Towards Robust Semantic Role Labeling

Sameer S. Pradhan
BBN Technologies

Wayne Ward
University of Colorado

James H. Martin
University of Colorado

Most Semantic Role labeling research has been focused on training and evaluating on the same corpus in order to develop the technology. This strategy, while appropriate for initiating research, can lead to over-training to the particular corpus. The new work presented in this paper focuses on analyzing the robustness of an SRL system when trained on one genre of data and used to label a different genre. We first describe the development of a state-of-the-art system for labeling propbank arguments on Wall Street Journal (WSJ) data. We detail further developments to improve performance on the WSJ corpus – including evaluating new features of several types including features from dependency parses, a features selection and calibration process, error analysis that pointed out the problem with the method of classifying constituents produced by a single syntactic parse, and an architecture for combining multiple syntactic views. While performing well on WSJ test data, such a system shows significant performance degradation when applied to label test data that is different than the genre of the data that it was trained on. We then present a series of experiments designed to investigate the source of this lack of portability. These experiments are based on comparisons of performance using PropBanked WSJ data and PropBanked Brown corpus data. Our results indicate that while syntactic parses and argument identification port relatively well to a new corpus, argument classification does not. Our analysis of the reasons for this is presented and generally point to the nature of the more lexical/semantic features dominating the classification task where more general structural features are more dominant in the argument identification task.

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 (Harabagiu, Bejan, and Morarescu 2005), Question Answering (Narayanan and Harabagiu 2004) and Summarization. Semantic role labeling 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. Using correct syntactic parses it is possible to achieve accuracies rivaling human inter-annotator agreement. More recent approaches have involved using improved features such as *n*-best parses (Koomen et

al. 2005; Toutanova, Haghighi, and Manning 2005; Haghighi, Toutanova, and Manning 2005), exploiting argument interdependence (Jiang, Li, and Ng 2005), using information from fundamentally different, and complementary syntactic views (Pradhan et al. 2005), combining hypotheses from different labeling systems using inference (Màrquez et al. 2005), as well as applied novel learning paradigms (Punyakanok et al. 2005; Toutanova, Haghighi, and Manning 2005; Moschitti 2006) that try to capture more sequence and contextual information. Some have also tried to jointly decode the syntactic and semantic structures (Yi and Palmer 2005; Musillo and Merlo 2006). In fact, this has been the subject of two CoNLL shared tasks (Carreras and Marquez 2004; Carreras and Màrquez 2005). While all these systems perform quite well on the WSJ test data, they show significant performance degradation when applied to label test data that is different than the genre of the data that it was trained on.

The new work presented in this paper focuses on analyzing the robustness of an SRL system when trained on one genre of data and used to label a different genre. We present a series of experiments designed to investigate the source of portability of our semantic role labeling system. These experiments are based on comparisons of performance using PropBanked WSJ data and PropBanked Brown corpus data. Our results indicate that while syntactic parses and argument identification port relatively well to a new corpus, argument classification does not. Our analysis of the reasons for this is presented and generally point to the nature of the more lexical/semantic features dominating the classification task where more general structural features are more dominant in the argument identification task.

The remainder of this paper is organized as follows. We first describe the development of a state-of-the-art baseline system for labeling propbank arguments on Wall Street Journal (WSJ) data. A detailed error analysis of our system indicates that the identification problem poses a significant bottleneck to improving overall system performance on WSJ. 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.

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 features, including features extracted from a parser that provides a different syntactic view – a Combinatory Categorial Grammar (CCG) parser (Hockenmaier and Steedman 2002b). 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), and another based on a flat, shallow syntactic chunk representation (Hacioglu 2004). We show that these two views complement each other to improve performance.

The second half of the paper starts by presenting initial evidence that such a system shows significant performance degradation when applied to label test data that is different than the genre of the data that it was trained on. Further, we present a series of experiments designed to investigate the source of this lack of portability. These

experiments are based on comparisons of performance using PropBanked WSJ data and PropBanked Brown corpus data.

2. Semantic Annotation and Corpora

We will be reporting on results using PropBank[1] (Palmer et al., 2005), a corpus in which predicate argument relations are marked for 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.

PropBank was constructed by assigning semantic arguments to constituents of the hand-corrected Treebank parses (hence Treebank parses.) Sometimes the tree can have *trace* nodes which refer to another node in the tree, but do not have any words associated with them. These can also be marked as arguments. As traces are not reproduced by standard syntactic parsers, we decided not to consider them in our experiments – whether or not they represent arguments of a predicate. PropBank also contains arguments that are coreferential and those that are discontinuous. We treat discontinuous and coreferential arguments in accordance to the CoNLL 2004/5 shared task on semantic role labeling. The first part of a discontinuous argument is labeled as it is, while the second part of the argument is labeled with a prefix "C-" appended to it. All coreferential arguments are labeled with a prefix "R-" appended to them.

The data comprise several sections of 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. The Feb 2004 release training set contains about 104,000 predicates instantiating about 250,000 arguments, and the test set comprises 5,400 predicates instantiating about 12,000 arguments.

3. Problem Description

The problem of shallow semantic role labeling can be viewed as three different tasks as introduced by Gildea and Jurafsky (2002).
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.

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

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

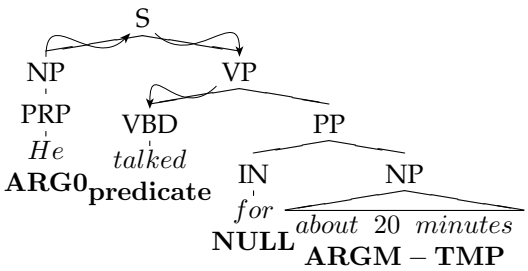

Figure 1
Syntax tree for a sentence illustrating the PropBank arguments.

4. ASSERT (Automatic Statistical SEmantic Role Tagger)

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 classification 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.

2 http://cl.aist-nara.ac.jp/~talus-Au/software/TinySVM/
3 http://cl.aist-nara.ac.jp/~taku-Au/software/yamcha/

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). Following is the list of features used:

1. **Predicate Lemma**
2. **Path –** Path from the constituent to the predicate in the parse tree.
3. **Position –** Whether the constituent is before or after the predicate.
4. **Voice**
5. **Predicate sub-categorization**
6. **Predicate Cluster**
7. **Head Word –** Head word of the constituent.
8. **Head Word POS –** POS of the head word
9. **Named Entities in Constituents –** 7 named entities as 7 binary features.
10. **Partial Path –** Path from the constituent to the lowest common ancestor of the predicate and the constituent.
11. **Verb Sense Information –** Oracle verb sense information from PropBank
12. **Head Word of PP –** Head of PP replaced by head word of NP inside it, and PP tag replaced by PP-*preposition*
13. **First and Last Word/POS in Constituent**
14. **Ordinal Constituent Position**
15. **Constituent Tree Distance**
16. **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.
17. **Temporal Cue Words**
18. **Dynamic Class Context**
19. **Syntactic Frame**
20. **Content Word Features –** Content word, its POS and named entities in the content word

As described in (Pradhan et al. 2004), 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. The search is constrained in such a way that no two NON-NULL nodes overlap with each other.

ALL ARGS	Task	P (%)	R (%)	F	A (%)
TREEBANK	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 1
Baseline system performance on all three tasks using Treebank parses and automatic parses on PropBank data.

Table 1 shows the performance of the system using the Treebank (TREEBANK) and using parses produced by a Charniak parser (AUTOMATIC). Precision (P), Recall (R) and F 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.

5. Additional Features

5.1 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 role 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

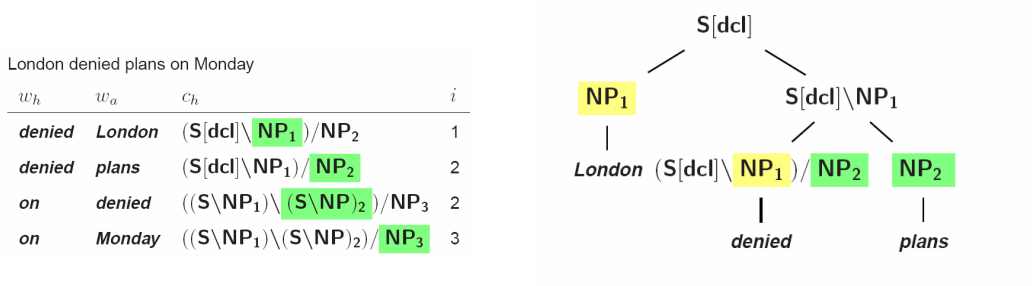

Figure 2
CCG parse of a sentence.

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 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 2002b). We incorporated these features in the systems based on 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 2 shows the performance of the system when these features were added. The corresponding baseline performances are mentioned in parentheses.

ALL ARGS	Task	P (%)	R (%)	F
TREEBANK	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 2
Performance improvement upon adding CCG features to the Baseline system.

5.2 Other Features

We added the following more generalizable features to the system:

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

5.2.1 Path Generalizations.

1. **Clause-based path variations –** Position of the clause node (S, SBAR) seems to be an important feature in argument identification (Hacioglu et al. 2004) Therefore we experimented with four clause-based path feature variations.
   (a) 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
   (b) Retaining only the clause nodes in the path, which for the above example would produce NP↑S↑S↓VBD,
   (c) Adding a binary feature that indicates whether the constituent is in the same clause as the predicate,
   (d) Collapsing the nodes between S nodes which gives NP↑S↑NP↑VP↓VBD.
2. **Path n-grams –** This feature decomposes a path into a series of trigrams. For example, the path NP↑S↑VP↑SBAR↑NP↑VP↓VBD becomes: NP↑S↑VP, S↑VP↑SBAR, VP↑SBAR↑NP, SBAR↑NP↑VP, etc. Shorter paths were padded with nulls.
3. **Single character phrase tags –** Each phrase category is clustered to a category defined by the first character of the phrase label.

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

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

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

5.3 N-Best Parses

Following (Koomen et al. 2005; Toutanova, Haghighi, and Manning 2005; Haghighi, Toutanova, and Manning 2005), we also now use *n*-best parse hypotheses, and consider the bag of constituents as potential argument candidates. We treat these as bag of constituents while performing the search through the argument lattice using argument sequence information.

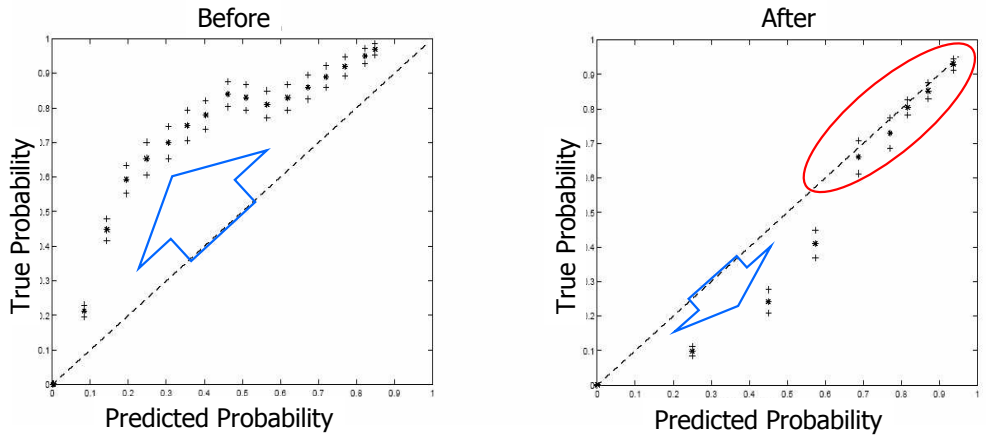

Figure 3
Plots showing true probabilities versus predicted probabilities before and after calibration on the test set for ARGM-TMP

.

6. 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 (2000) to convert the SVM scores into probabilities by fitting to a sigmoid. When all classifiers used the same set of features, fitting all scores to a single sigmoid was found to give the best performance. Since different feature sets are now used by the classifiers, we trained a separate sigmoid for each classifier.

Foster and Stine (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

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

Table 3
Performance improvement on selecting features per argument and calibrating the probabilities on 10k training data on WSJ section 23.

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 3. The performance improvement over a classifier that is trained using all the features for all the classes is shown in Table 3.

Table 4 shows the performance of the system after adding the CCG features, additional features extracted from the Charniak parse tree, and performing feature selection and calibration. Numbers in parentheses are the corresponding baseline performances.

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

Table 4
Best system performance on all tasks using automatically generated syntactic parses.

7. 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 one additional representations: i) chunking parser (Hacioglu et al. 2004). The hope is that it will produce different errors than the Charniak parser since it represents a different syntactic view. The Charniak parser is trained on the Penn Treebank corpus. 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 two representations could produce better results than any single one.

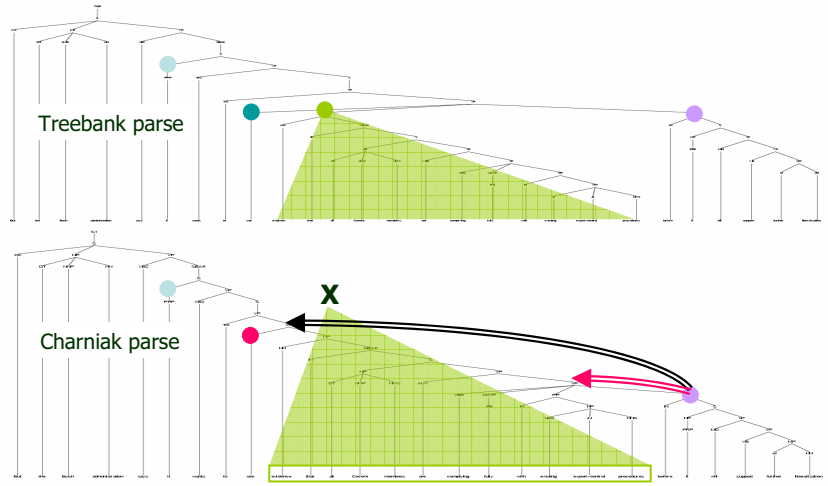

Figure 4
Illustration of how a parse error affects argument identification.

7.1 Chunk-based Semantic Role Labeler

Hacioglu has previously described a chunk based semantic role 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 roles to the chunks. The system is trained on the PropBank training data. Following features are used to train the base semantic chunker:

1. **Words**
2. **Predicate lemmas**
3. **Part of Speech tags**
4. **BP Positions –** The position of a token in a BP using the IOB2 representation (e.g. B-NP, I-NP, O, etc.)
5. **Clause tags –** The tags that mark token positions in a sentence with respect to clauses.
6. **Named entities –** The IOB tags of named entities.

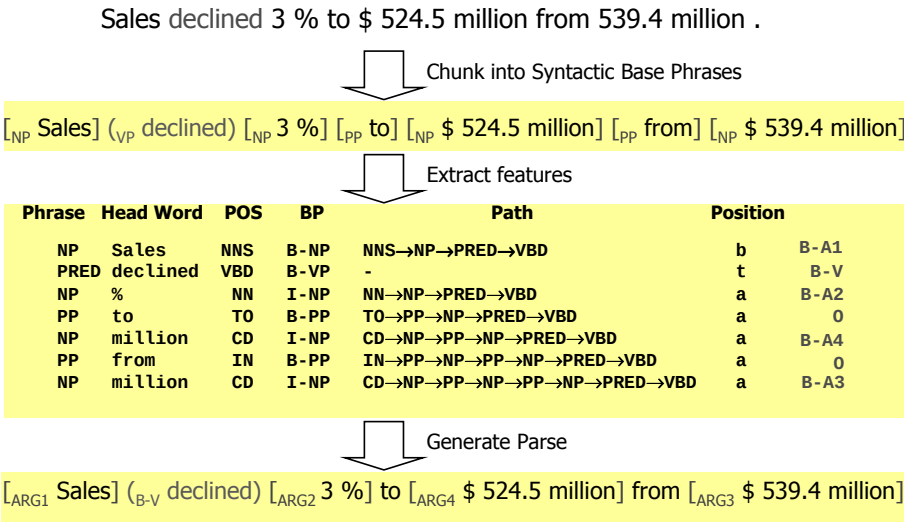

Figure 5
Semantic Chunker.

7. **Token Position –** The position of the phrase with respect to the predicate. It has three values as "before", "after" and "-" (for the predicate)
8. **Path –** It defines a flat path between the token and the predicate
9. **Clause bracket patterns**
10. **Clause position –** A binary feature that identifies whether the token is inside or outside the clause containing the predicate
11. **Headword suffixes –** Suffixes of headwords of length 2, 3 and 4.
12. **Distance –** Distance of the token from the predicate as a number of base phrases, and the distance as the number of VP chunks.
13. **Length –** The number of words in a token.
14. **Predicate POS tag –** The part of speech category of the predicate
15. **Predicate Frequency –** Frequent or rare using a threshold of 3.
16. **Predicate BP Context –** The chain of BPs centered at the predicate within a window of size -2/+2.
17. **Predicate POS context –** POS tags of words immediately preceding and following the predicate.
18. **Predicate Argument Frames –** Left and right core argument patterns around the predicate.
19. **Number of predicates –** This is the number of predicates in the sentence.

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 roles 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 5
Semantic role 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, 2001) are used as the SVM training and test software.

Table 5 presents the system performances on the PropBank test set for the chunk-based system.

We combined the output of semantic role labelers 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 6 shows the performance improvement after the combination. Again, numbers in parentheses are respective baseline performances.

TASK	P (%)	R (%)	F
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 6
Constituent-based best system performance on argument identification and argument identification and classification tasks after combining both semantic role labelers.

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. 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, Chunk hypotheses then that was selected. The results for this are shown in Table 7. It can be seen that the head word based performance almost approaches the constituent based performance reported on the Treebank parses in Table 2 and there seems to be considerable scope for improvement.

5 http://chasen.org/~taku/software/TinySVM/
6 http://chasen.org/~taku/software/yamcha/

	Task	P (%)	R (%)	F
C	Id.	92.2	87.5	89.8
	Id. + Classification	85.9	81.6	83.7
C+CH	Id.	98.9	88.8	93.6
	Id. + Classification	92.5	83.3	87.7

Table 7
Performance improvement on head word based scoring after oracle combination. Charniak (C), and Chunker (CH).

Table 8 shows the performance improvement in the actual system after combination.

	Task	P (%)	R (%)	F
C	Id.	92.2	87.5	89.8
	Id. + Classification	85.9	81.6	83.7
C+CH	Id.	91.5	91.1	91.3
	Id. + Classification	84.9	84.3	84.7

Table 8
Performance improvement on head word based scoring after combination. Charniak (C) and Chunker (CH).

8. Improved Architecture

The attempt to combine the hypotheses generated by the Charniak based semantic role labeler and Chunk based semantic role labeler in the preceding section, after the fact, seemed to be suboptimal, so now, we propose what we believe is an improved framework for combining information from different syntactic views. Our goal is to preserve the robustness and flexibility of the segmentation of the phrase-based chunker, but to take advantage of features from full syntactic parses. We also want to combine features from different syntactic parses to gain additional robustness.

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

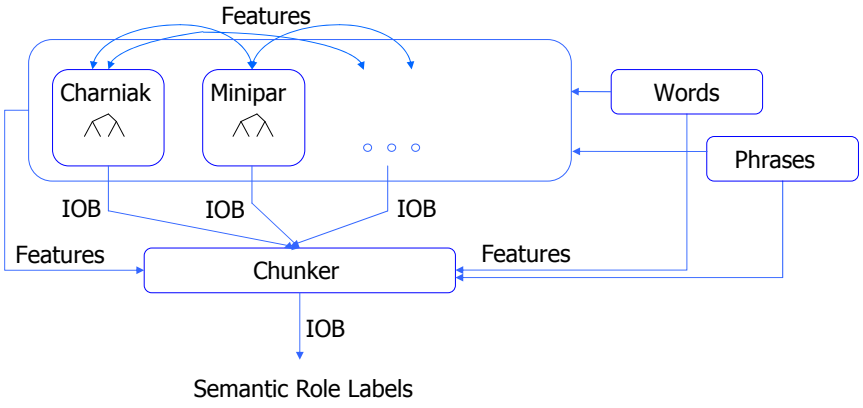

Figure 6
New Architecture.

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

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

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

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

SVMs were trained for begin (B) and inside (I) classes of all arguments and an outside (O) class. One particular advantage of this architecture, as dipicted in Figure 7 is that the final segmentation does not have to necessarily be adhering to one of the input segmentations, and depending on the provided information in terms of features, the classifier can generate a new, better segmentation.

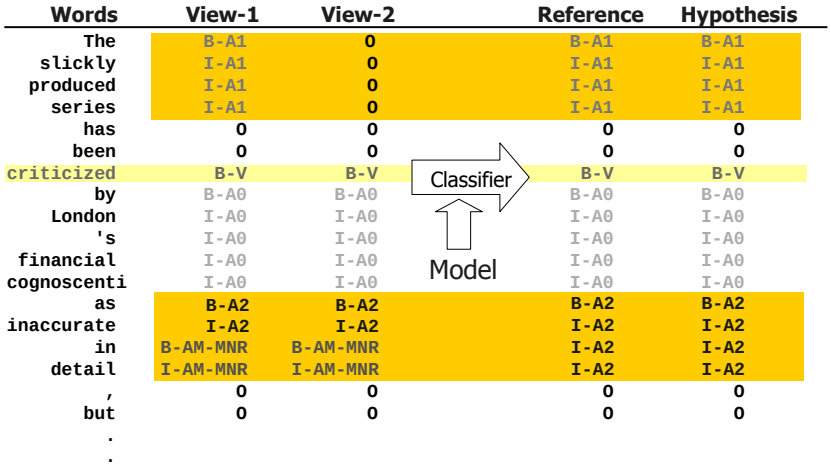

Figure 7
Example classification using the new architecture.

We participated in the CoNLL shared task on SRL (Carreras and Màrquez 2005). This gave us an opportunity our integrated architecture. For this evaluation, various syntactic features were provided, including output of a Base Phrase chunker, Charniak parser, Collins' parser, Clause tagger and Named Entities. In this evaluation, in addition to the WSJ section-23, a portion of a completely different genre of text from The Brown Corpus (Kučera and Francis 1967) was used. The section of the Brown Corpus that was used for evaluation contains prose from "general fiction" and represents quite a different genre of material from what the SRL systems were trained on. This was done to investigate into the robustness of these systems.

Table 9 shows the performance of the new architecture on the CoNLL 2005 test set. As it can be seen, almost all the state-of-the-art systems – including the one described so far, suffered a 10 point drop in the F measure.

9. Robustness to Genre of Data

So far most of the recent work on SRL systems has been focused on improving the labeling performance on a test set belonging to the same genre of text as the training set. Both, the Treebank on which the syntactic parser is trained, and the PropBank on which the SRL systems are trained represent articles from the year 1989 of the Wall Street Journal. Part of the reason for this being the availability of data tagged with

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

Table 9
Performance of the integrated architecture on the CoNLL-2005 shared task on semantic role labeling.

similar semantic argument structure in multiple genres of text. At this juncture it is quite possible that these tools are being subject to the effects of over-training to this genre of text, and instead of any improvements to the system reflecting further progress in the field, are getting tuned to this style of journalistic text. It is also important for this technology to be widely accepted that it performs reasonably well on text that does not represent the Wall Street Journal.

As a pilot experiment, Pradhan et al. (2004, 2005) reported some preliminary analysis on a small test set that was tagged for a small portion of about 400 sentences from the AQUAINT corpus, which is a collection of articles from the New York Times, Xinhua, and FBIS. Although this collection also represent newswire text, it was aimed at finding out whether an incremental change within the same general domain that occurred in a different time period and presumably representing different entities and events and maybe a different style would impact the performance of the SRL system. As a matter of fact, the system performance dropped from F-scores in the high 70s to low 60s. As we now know, this is not much different from the performance obtained on a test set from the Brown corpus which represents quite different style of text. Fortunately, Palmer, Gildea, and Kingsbury (2005) have also recently PropBanked a significant portion of the Brown corpus, and therefore it is possible to perform a more systematic analysis of the portability of SRL systems from one genre of text to another.

9.1 The Brown Corpus

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

A. Press Reportage

B. Press Editorial

C. Press Reviews (theater, books, music and dance)

D. Religion

E. Skills and Hobbies

F. Popular Lore

G. Belles Lettres, Biography, Memoirs, etc.

H. Miscellaneous
J. Learned
K. General Fiction
L. Mystery and Detective Fiction
M. Science Fiction
N. Adventure and Western Fiction
P. Romance and Love Story
R. Humor

9.2 Semantic Annotation

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

Table 10 gives the amount of predicates that have been tagged from each section:

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

Table 10
Number of predicates that have been tagged in the PropBanked portion of Brown corpus

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

10. Robustness Experiments

This section focuses on various experiments that we performed on the PropBanked Brown corpus and which could go some way in analyzing the factors that affect the portability of SRL systems, and might throw some light on what steps need to be taken to improving the same. In order to avoid confounding the effects of the two distinct

views that we saw before – the top-down syntactic view that extracts features from a syntactic parse, and the bottom-up phrase chunking view, for all these experiments – except one, we will use the system that is based on classifying constituents in a syntactic tree with PropBank arguments.

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

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

The test set that was used for the CoNLL evaluation was a part of one of the Brown sections consisting of about 800 examples from the section CK. This is about 5% of the available PropBank Brown arguments, so we decided to use the entire Brown corpus as a test set for this experiment and use ASSERT trained on WSJ sections 02-21 to tag its arguments.

10.1.1 Results. Table 11 gives the details of the performance over each of the eight different text genres. It can be seen that on an average, the F-score on the combined task of identification and classification is comparable to the ones obtained on the AQUAINT test set. It is interesting to note that although AQUAINT is a different text source, it is still essentially newswire text. However, even though Brown corpus has much more variety, on an average, the degradation in performance is almost identical. This tells us that maybe the models are tuned to the particular vocabulary and sense structure associated with the training data. Also, since the syntactic parser that is used for generating the parse trees is also heavily lexicalized, could also have some impact on the accuracy of the parses, and the features extracted from them.

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

Table 11
Performance on the entire PropBanked Brown corpus.

In order to check the extent of the deletion errors owing to the parser mistakes which result in the constituents representing a valid node getting deleted, we generated the appropriate numbers which are shown in Table 12. These numbers are for top one parse.

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

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

Table 12
Constituent deletions in WSJ test set and the entire PropBanked Brown corpus.

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

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

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

In this experiment, we wanted to remove the effect of errors in estimating the syntactic structure. Therefore, we used correct syntactic trees from the Treebank. We trained ASSERT on a Brown training set and tested it on a test set also from the Brown corpus. Instead of using the CoNLL test set which represents part of section CK, we decided to use a stratified test set as used by the syntactic parsing community (Gildea 2001). The test set is generated by selecting every 10^{th} sentence in the Brown Corpus. We also held out a development set used by Bacchiani et al. (2006) to tune system parameters for in the future. We did not perform any parameter tuning specially for this or any of the following experiments, and used the same parameters as that reported for the best performing version of ASSERT as reported in Table4 of this article. We compare the performance on this test set with that obtained when ASSERT is trained using WSJ sections 00-21 and use section 23 for testing. For a more balanced comparison, we also retrained ASSERT on the same amount of data as used for training it on Brown, and tested it on section 23. As usual, trace information, and function tag information from the Treebank is stripped out.

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

Table 13
Performance when ASSERT is trained using correct Treebank parses, and is used to classify test set from either the same genre or another. For each dataset, the number of examples used for training are shown in parenthesis

10.2.1 Results. Table 13 shows that there is a very negligible difference in argument identification performance when ASSERT is trained on 14,000 predicates and 104,000 predicates from the WSJ. We can notice a considerable drop in classification accuracy though. Further, when ASSERT is trained on Brown training data and tested on the Brown test data, the argument identification performance is quite similar to the one that is obtained on the WSJ test set using ASSERT trained on Treebank WSJ parses. It tells us that the drop in argument classification accuracy is much more severe. We know that the predicate whose arguments are being identified, and the head word of the syntactic constituent being classified are both important features in the task of argument classification. This evidence tends to indicate one of the following: i) maybe the task of classification needs much more data to train, and that this is merely an effect of the quantity of data, ii) maybe the predicates and head words (or, words in general) in a homogeneous corpus such as the WSJ are used more consistently, and that the style is simple and therefore it becomes an easier task for classification as opposed to the various usages and senses in a heterogeneous collection such as the Brown corpus, iii) the features that are used for classification are more appropriate for WSJ than for Brown.

10.3 Experiment 3: How much does correct structure help?

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

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

10.3.1 Results. Table 13 shows that the syntactic information from WSJ transfers quite well to the Brown corpus. Once again we see, that there is a very slight drop in argument identification performance, but an even greater drop in the argument classification accuracy.

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

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

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

Recently there have been some improvements to the Charniak parser, and that provides us with an opportunity to experiment with its latest version that does *n*-best re-ranking as reported in (Charniak and Johnson 2005) and one that uses self-training and re-ranking using data from the North American News corpus (NANC) and adapts much better to the Brown corpus (McClosky, Charniak, and Johnson 2006a, 2006b). We also use another one that is trained on Brown corpus itself. The performance of these parsers as reported in the respective literature are shown in Table 14

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

Table 14
Performance of different versions of Charniak parser used in the experiments.

We describe the results of the following five experiments:

1. ASSERT is trained on features extracted from automatically generated parses of the PropBanked WSJ sentences. The syntactic parser – Charniak parser – is itself trained on the WSJ training sections of the Treebank. This is used to classify the section-23 of WSJ.
2. ASSERT is trained on features extracted from automatically generated parses of the PropBanked WSJ sentences. The syntactic parser – Charniak parser – is itself trained on the WSJ training sections of the Treebank. This is used to classify the Brown test set.
3. ASSERT is trained on features extracted from automatically generated parses of the PropBanked Brown corpus sentences. The syntactic parser is trained using the WSJ portion of the Treebank. This is used to classify the Brown test set.
4. ASSERT is trained on features extracted from automatically generated parses of the PropBanked Brown corpus sentences. The syntactic parser is

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

Table 15
Performance on WSJ and Brown test set when ASSERT is trained on features extracted from automatically generated syntactic parses

trained using the Brown training portion of the Treebank. This is used to classify the Brown test set.

5. ASSERT is trained on features extracted from automatically generated parses of the PropBanked Brown corpus sentences. The syntactic parser is the version that is self-trained using 2,500,000 sentences from NANC, and where the starting version is trained only on WSJ data (McClosky, Charniak, and Johnson 2006b). This is used to classify the Brown test set.

10.4.1 Results. Table 15 shows the results of these experiments. For simplicity of discussion we have tagged the five setups as A., B., C., D., and E. Looking at setups B. and C. it can be seen that when the features used to train ASSERT are extracted using a syntactic parser that is trained on WSJ it performs at almost the same level on the task of identification, regardless of whether it is trained on the PropBanked Brown corpus or the PropBanked WSJ corpus. This, however, is about 5-6 F-score points lower than when all the three – the syntactic parser training set, ASSERT training set, and ASSERT test set, are from the same genre – WSJ or Brown, as seen in A. and D. In case of the combined task the gap between the performance for set up B. and C. is about 10 points F-score apart (59.1 vs 69.8) Looking at the argument classification accuracies, we see that using a ASSERT trained on WSJ to test Brown sentences give a 12 point drop in F-score. Using ASSERT trained on Brown using WSJ trained syntactic parser seems to drop in accuracy by about 5 F-score points. When ASSERT is trained on Brown using syntactic parser also trained on Brown, we get a quite similar classification performance, which is again about 5 points lower than what we get using all WSJ data. This shows lexical semantic features might be very important to get a better argument classification on Brown corpus.

Parser Train	BP Chunker Train	SRL Train	P (%)	R (%)	F
WSJ	WSJ	Brown	64.2	57.0	60.3
WSJ+NANC	WSJ	Brown	77.7	66.0	71.3

Table 16
Performance of the task of argument identification and classification using architecture that combines top down syntactic parses with flat syntactic chunks.

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

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

For this experiment we used the Syntactic parser trained on WSJ and one that is adapted through self-training using the NANC, and a base phrase chunker that is trained on WSJ Treebank, and use the integrated architecture as described in Section 8.

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

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

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

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

10.6.1 Results. Table 17 shows the result of these experiments. It can be seen that in all the six settings, the performance on the task of identification and classification improves gradually until about 5625 examples of section CK which is about 75% of the total added, above which it adds very little. It is very nice to note that even when the syntactic parser is trained on WSJ and the SRL is trained on WSJ, that adding 7,500 instances of this new genres allows it to achieve almost the same amount of performance as that achieved when all the three are from the same genre (67.2 vs 69.9) As for the task of argument identification, the incremental addition of data from the new genre shows

only minimally improvement. The system that uses self-trained syntactic parser seems to perform slightly better than the rest of the versions that use automatically generated syntactic parses. Another point that might be worth noting is that the improvement on the identification performance is almost exclusively to the recall. The precison number are almost unaffected – except when the labeler is trained on WSJ PropBank data.

Parser	SRL	Id.			Id. + Class		
		P	R	F	P	R	F
Train	Train	(%)	(%)		(%)	(%)	
WSJ	WSJ (14k) (Treebank parses)						
(Treebank parses)	+0 examples from CK	96.2	91.9	94.0	74.1	66.5	70.1
	+1875 examples from CK	96.1	92.9	94.5	77.6	71.3	74.3
	+3750 examples from CK	96.3	94.2	95.1	79.1	74.1	76.5
	+5625 examples from CK	96.4	94.8	95.6	80.4	76.1	78.1
	+7500 examples from CK	96.4	95.2	95.8	80.2	76.1	78.1
Brown	Brown (14k) (Treebank parses)						
(Treebank parses)	+0 examples from CK	96.1	94.2	95.1	77.1	73.0	75.0
	+1875 examples from CK	96.1	95.4	95.7	78.8	75.1	76.9
	+3750 examples from CK	96.3	94.6	95.3	80.4	76.9	78.6
	+5625 examples from CK	96.2	94.8	95.5	80.4	77.2	78.7
	+7500 examples from CK	96.3	95.1	95.7	81.2	78.1	79.6
WSJ	WSJ (14k)						
(40k)	+0 examples from CK	83.1	78.8	80.9	65.2	55.7	60.1
	+1875 examples from CK	83.4	79.3	81.3	68.9	57.5	62.7
	+3750 examples from CK	83.9	79.1	81.4	71.8	59.3	64.9
	+5625 examples from CK	84.5	79.5	81.9	74.3	61.3	67.2
	+7500 examples from CK	84.8	79.4	82.0	74.8	61.0	67.2
WSJ	Brown (14k)						
(40k)	+0 examples from CK	85.7	77.2	81.2	74.4	57.0	64.5
	+1875 examples from CK	85.7	77.6	81.4	75.1	58.7	65.9
	+3750 examples from CK	85.6	78.1	81.7	76.1	59.6	66.9
	+5625 examples from CK	85.7	78.5	81.9	76.9	60.5	67.7
	+7500 examples from CK	85.9	78.1	81.7	76.8	59.8	67.2
Brown	Brown (14k)						
(20k)	+0 examples from CK	87.6	80.6	83.9	76.0	59.2	66.5
	+1875 examples from CK	87.4	81.2	84.1	76.1	60.0	67.1
	+3750 examples from CK	87.5	81.6	84.4	77.7	62.4	69.2
	+5625 examples from CK	87.5	82.0	84.6	78.2	63.5	70.1
	+7500 examples from CK	87.3	82.1	84.6	78.2	63.2	69.9
WSJ+NANC	Brown (14k)						
(2,500k)	+0 examples from CK	89.1	81.7	85.2	74.4	60.1	66.5
	+1875 examples from CK	88.6	82.2	85.2	76.2	62.3	68.5
	+3750 examples from CK	88.3	82.6	85.3	76.8	63.6	69.6
	+5625 examples from CK	88.3	82.4	85.2	77.7	63.8	70.0
	+7500 examples from CK	88.9	82.9	85.8	78.2	64.9	70.9

Table 17
Effect of incrementally adding data from a new genre

11. General Discussion

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

(1) SRC enterprise prevented John from [$_{\mathbf{predicate}}$ taking] [$_{\mathbf{ARG1}}$ the assignment]

(2) SRC enterprise prevented [$_{\mathbf{ARG0}}$ John] from [$_{\mathbf{predicate}}$ selling] [$_{\mathbf{ARG1}}$ the assignment]

In example (1), "John" is not marked as the ARG0 of "taking" Whereas in example (2), replacing the predicate "taking" with "selling" corrects the semantic roles, even though the syntactic parse for both sentences is *exactly* the same. Even using several other predicates in place of "taking" such as "distributing," "submitting," etc. give a correct parse. So there is some idiosyncrasy with the predicate "take"

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

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

(4) [$_{\mathbf{ARG1}}$ The stock] [$_{\mathbf{predicate}}$ jumped] [$_{\mathbf{ARG4}}$ to $ 140 billion from $ 250 billion in a few hours of time]

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

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

(7) [$_{\mathbf{ARG1}}$ The stock] [$_{\mathbf{predicate}}$ jumped] [$_{\mathbf{ARG4}}$ to $ 140 billion] [$_{\mathbf{ARG3}}$ from $ 250 billion] [$_{\mathbf{ARGM-TMP}}$ yesterday]

(8) [$_{\mathbf{ARG1}}$ The stock] [$_{\mathbf{predicate}}$ increased] [$_{\mathbf{ARG4}}$ to $ 140 billion] [$_{\mathbf{ARG3}}$ from $ 250 billion] [$_{\mathbf{ARGM-TMP}}$ yesterday]

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

(10) [$_{\mathbf{ARG1}}$ The stock] [$_{\mathbf{predicate}}$ dropped] [$_{\mathbf{ARG4}}$ to $ 140 billion] [$_{\mathbf{ARG3}}$ from $ 250 billion within a few hours]

WSJ articles almost always report jump in stock prices by the phrase "to .." followed by "from ..." and the syntactic parser statistics seem to be tuned to that, and therefore when it faces a sentence like the example (3) above, two sibling noun phrases are collapsed into one phrase, and so the there is only one node in the tree for the two different arguments ARG3 and ARG4 and therefore the role labeler tags it as the more probable of the two and that being ARG3. In example (4), the two noun phrases are identified correctly. The difference in the two is just the transposition of the two words

"to" and "from" In the example (4), however, the prepositional phrase "in a few hours of time" get attached to the wrong node in the tree, and therefore deleting the node that would have identified the exact boundary of the second argument. Upon deleting the part of the text that is the wrongly attached prepositional phrase, we get the correct semantic roles in example (5). Now, lets replace this prepositional phrase with a string that happens to be present in the WSJ training data, and see what happens. As seen in example (6), the parser identifies and attaches this phrase correctly and we get a completely correct set of tags. This further strengthens our claim. Even replacing the temporal with a simple one such as "yesterday" maintains the correctness of the tags and also replacing "jumped" with "increased" maintains its correctness. Now, lets see what happens when the predicate "jump" in example (4) is changed to yet another synonymous predicate – "dropped". Doing this gives us a correct tagset even though the same syntactic structure is shared between the two, and the prepositional phrase was not attached properly earlier. This shows that just the change of a verb to another changes the syntactic parse to align with the right semantic interpretation. Changing the temporal argument to something slightly different once again causes the parse to fail as seen in (10).

12. Conclusions

Our experimental results on robustness to change in genre can be summarized as follows:

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

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

1. **Lexical cues –** word usage specific to WSJ.
2. **Verb sub-categorizations –** They can vary considerably from one sample of text to another as seen in the examples above and as evaluated in an empirical study by (Roland and Jurafsky 1998)
3. **Word senses –** domination by unusual word senses (stocks fell)
4. **Topics and entities**

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

1. **Less homogeneous corpora –** Rather than using many examples drawn from one source, fewer examples could be drawn from many sources. This would reduce the likelihood of learning idiosyncratic senses and argument structures for predicates.
2. **Less specific entities –** Entity values could be replaced by their class tag (person, organization, location, etc). This would reduce the likelihood of learning idiosyncratic associations between specific entities and predicates. The system could be forced to use this and more general features.

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

13. Acknowledgments

We are extremely grateful to Martha Palmer for providing us with the PropBanked Brown corpus, and to David McClosky for providing us with hypotheses on the Brown test set as well as a cross-validated version of the Brown training data for the various models reported in his work reported at HLT 2006.

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.

Special thanks to Matthew Woitaszek, Theron Voran and the other administrative team of the Hemisphere and Occam Beowulf clusters. Without these the training would never be possible.

References

Bacchiani, Michiel, Michael Riley, Brian Roark, and Richard Sproat. 2006. MAP adaptation of stochastic grammars. *Computer Speech and Language*, 20(1):41–68.

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

Carreras, Xavier and Lluis Marquez. 2004. Introduction to the conll-2004 shared task: Semantic role labeling. In *Proceedings of 8th Conference on CoNLL-2004*.

Carreras, Xavier and Lluís Màrquez. 2005. Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 152–164, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Charniak, Eugene. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st Annual Meeting of the North American Chapter of the ACL (NAACL)*, pages 132–139, Seattle, Washington.

Charniak, Eugene and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 173–180, Ann Arbor, Michigan, June. Association for Computational Linguistics.

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

Gildea, Dan and Julia Hockenmaier. 2003. Identifying semantic roles using combinatory categorial grammar. In *Proceedings of the Conference on Empirical Methods in Natural Language*

Processing, Sapporo, Japan.
Gildea, Daniel. 2001. Corpus variation and parser performance. In *In Proceedings of Empirical Methors in Natural Language Processing (EMNLP)*.
Gildea, Daniel and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.
Hacioglu, Kadri. 2004. A lightweight semantic chunking model based on tagging. In *Proceedings of the Human Language Technology Conference /North American chapter of the Association of Computational Linguistics (HLT/NAACL)*, Boston, MA.
Hacioglu, Kadri, 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.
Hacioglu, Kadri, Sameer Pradhan, Wayne Ward, James Martin, and Daniel Jurafsky. 2004. Semantic role labeling by tagging syntactic chunks. In *Proceedings of the 8th Conference on CoNLL-2004, Shared Task – Semantic Role Labeling*.
Haghighi, Aria, Kristina Toutanova, and Christopher Manning. 2005. A joint model for semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 173–176, Ann Arbor, Michigan, June. Association for Computational Linguistics.
Harabagiu, Sanda, Cosmin Adrian Bejan, and Paul Morarescu. 2005. Shallow semantics for relation extraction. In *Nineteenth International Joint Conference on Artificial Intelligence*, pages 1061–1067, Edinburgh, Scotland, August.
Hockenmaier, Julia and Mark Steedman. 2002a. Acquiring compact lexicalized grammars from a cleaner treebank.
Hockenmaier, Julia and Mark Steedman. 2002b. Generative models for statistical parsing with combinatory grammars. In *Proceedings of the 40th meeting of the ACL*, pages 335–342.
Jiang, Zheng Ping, Jia Li, and Hwee Tou Ng. 2005. Semantic argument classification exploting argument interdependence. In *Nineteenth International Joint Conference on Artificial Intelligence*, pages 1067–1073, Edinburgh, Scotland, August.
Koomen, Peter, Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2005. Generalized inference with multiple semantic role labeling systems. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 181–184, Ann Arbor, Michigan, June. Association for Computational Linguistics.
Kučera, Henry and W. Nelson Francis. 1967. *Computational analysis of present-day American English*. Brown University Press, Providence, RI.
Kudo, Taku and Yuji Matsumoto. 2000. Use of support vector learning for chunk identification. In *Proceedings of the 4th Conference on CoNLL-2000 and LLL-2000*, pages 142–144.
Kudo, Taku and Yuji Matsumoto. 2001. Chunking with support vector machines. In *Proceedings of the 2nd Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL-2001)*.
Marcus, Mitchell, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The Penn Treebank: Annotating predicate argument structure.
Màrquez, Lluís, Mihai Surdeanu, Pere Comas, and Jordi Turmo. 2005. A robust combination strategy for semantic role labeling. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 644–651, Vancouver, British Columbia, Canada, October. Association for Computational Linguistics.
McClosky, David, Eugene Charniak, and Mark Johnson. 2006a. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 152–159, New York City, USA, June. Association for Computational Linguistics.
McClosky, David, Eugene Charniak, and Mark Johnson. 2006b. Rerankinng and self-training for parser adaptation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (COLING-ACL'06)*, Sydney, Australia, July. Association for Computational Linguistics.
Moschitti, Alessandro. 2006. Syntactic kernels for natural language learning: the semantic role labeling case. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 97–100, New York City, USA, June. Association for Computational Linguistics.
Musillo, Gabriele and Paola Merlo. 2006. Accurate parsing of the proposition bank. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short*

Papers, pages 101–104, New York City, USA, June. Association for Computational Linguistics.
Narayanan, Srini and Sanda Harabagiu. 2004. Question answering based on semantic structures. In *Proceedings of the International Conference on Computational Linguistics (COLING '04)*, Geneva, Switzerland, August. Association for Computational Linguistics.
Palmer, Martha, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.
Platt, John. 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.
Pradhan, Sameer, Kadri Hacioglu, Valerie Krugler, Wayne Ward, James Martin, and Dan Jurafsky. 2005. Support vector learning for semantic argument classification. *Machine Learning Journal*, 60(1):11–39.
Pradhan, Sameer, Kadri Hacioglu, Wayne Ward, James Martin, and Dan Jurafsky. 2003. Semantic role parsing: Adding semantic structure to unstructured text. In *Proceedings of the International Conference on Data Mining (ICDM 2003)*, Melbourne, Florida.
Pradhan, Sameer, Wayne Ward, Kadri Hacioglu, James Martin, and Dan Jurafsky. 2004. Shallow semantic parsing using support vector machines. In *Proceedings of the Human Language Technology Conference/North American chapter of the Association of Computational Linguistics (HLT/NAACL)*, Boston, MA.
Pradhan, Sameer, Wayne Ward, Kadri Hacioglu, James Martin, and Dan Jurafsky. 2005. Semantic role labeling using different syntactic views. In *Proceedings of the Association for Computational Linguistics 43rd annual meeting (ACL-2005)*, Ann Arbor, MI.
Punyakanok, Vasin, Dan Roth, Wen tau Yih, and Dav Zimak. 2005. Learning and inference over constrained output. In *Nineteenth International Joint Conference on Artificial Intelligence*, pages 1124–1130, Edinburgh, Scotland, August.
Ramshaw, L. A. and M. P. Marcus. 1995. Text chunking using transformation-based learning. In *Proceedings of the Third Annual Workshop on Very Large Corpora*, pages 82–94. ACL.
Roland, Douglas and Daniel Jurafsky. 1998. How verb subcategorization frequencies are affected by corpus choice. In *Proceedings of COLING/ACL*, pages 1122–1128, Montreal, Canada.
Surdeanu, Mihai, Sanda Harabagiu, John Williams, and Paul Aarseth. 2003. Using predicate-argument structures for information extraction. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, Sapporo, Japan.
Toutanova, Kristina, Aria Haghighi, and Christopher Manning. 2005. Joint learning improves semantic role labeling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 589–596, Ann Arbor, Michigan, June. Association for Computational Linguistics.
Xue, Nianwen and Martha Palmer. 2004. Calibrating features for semantic role labeling. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Barcelona, Spain.
Yi, Szu-ting and Martha Palmer. 2005. The integration of syntactic parsing and semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 237–240, Ann Arbor, Michigan, June. Association for Computational Linguistics.