Semantic Argument Classification Exploiting Argument Interdependence

Zheng Ping Jiang [1] and **Jia Li** [2] and **Hwee Tou Ng** [1,2]
1. Singapore-MIT Alliance, E4-04-10, 4 Engineering Drive 3, Singapore 117576
2. Department of Computer Science, National University of Singapore,
3 Science Drive 2, Singapore 117543
{jiangzp, lijia1, nght}@comp.nus.edu.sg

Abstract

This paper describes our research on automatic semantic argument classification, using the PropBank data [Kingsbury *et al.*, 2002]. Previous research employed features that were based either on a full parse or shallow parse of a sentence. These features were mostly based on an individual semantic argument and the relation between the predicate and a semantic argument, but they did not capture the interdependence among all arguments of a predicate. In this paper, we propose the use of the neighboring semantic arguments of a predicate as additional features in determining the class of the current semantic argument. Our experimental results show significant improvement in the accuracy of semantic argument classification after exploiting argument interdependence. Argument classification accuracy on the standard Section 23 test set improves to 90.50%, representing a relative error reduction of 18%.

1 Introduction

Deriving the semantic representation of a sentence is important in many natural language processing tasks, such as information extraction and question answering. The recent availability of semantically annotated corpora, such as FrameNet [Baker *et al.*, 1998][1] and PropBank [Kingsbury *et al.*, 2002][2], prompted research in automatically producing the semantic representations of English sentences. For PropBank, the semantic representation annotated is in the form of semantic roles, such as $ARG0$, $ARG1$, etc. of each predicate in a sentence, and the process of determining these semantic roles is known as semantic role labeling.

Most previous research treats the semantic role labeling task as a classification problem, and divides the task into two subtasks: *semantic argument identification* and *semantic argument classification*. Semantic argument identification involves classifying each syntactic element in a sentence into either a semantic argument or a non-argument. A syntactic element can be a word in a sentence, a chunk in a shallow parse, or a constituent node in a full parse tree. Semantic argument classification involves classifying each semantic argument identified into a specific semantic role, such as $ARG0$, $ARG1$, etc.

Various features based on either a shallow parse or full parse of a sentence have been proposed. These features are then used by some machine learning algorithm to build a classifier for semantic argument classification. While these features capture the syntactic environment of the semantic argument currently being classified, they overlook the semantic context of the argument.

We propose a notion of *semantic context*, which consists of the already identified or role-classified semantic arguments in the context of an argument that is currently being classified. Semantic context features are defined as features extracted from the neighboring semantic arguments, and used in classifying the current semantic argument. The use of these features explicitly captures the interdependence among the semantic arguments of a predicate.

In this paper, we focus on the semantic argument classification task using PropBank[Kingsbury *et al.*, 2002]. Our semantic context features are derived from full parse trees. Our experimental results show significant improvement in the accuracy of semantic argument classification after adding the semantic context features, when compared to using only features of the current semantic argument.

The rest of this paper is organized as follows: Section 2 explains in detail the application of semantic context features to semantic argument classification; Section 3 gives a specific example to illustrate how semantic context features can improve semantic argument classification; Section 4 presents our experimental results; Section 5 reviews related work; and Section 6 concludes the paper.

2 Semantic Argument Classification with Semantic Context Features

Similar to [Pradhan *et al.*, 2005], we divide the task of semantic role labeling into two sequential subtasks: semantic argument identification and semantic argument classification, and treat each subtask as a separate classification problem. Our investigation focuses on semantic argument classification with the assumption that the semantic arguments have been correctly identified by the semantic argument identifica-

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

Feature	Description
Sentence level features	
predicate (Pr)	predicate lemma in the predicate-argument structure
voice (Vo)	grammatical voice of the predicate, either active or passive
subcat (Sc)	grammar rule that expands the predicate's parent node in the parse tree
Argument-specifi c features	
phrase type (Pt)	syntactic category of the argument constituent
head word (Hw)	head word of the argument constituent
Argument-predicate relational features	
path (Pa)	syntactic path through the parse tree from the argument constituent to the predicate node
position (Po)	relative position of the argument consitituent with respect to the predicate node, either left or right

Table 1: Baseline features

tion module in the first step.

2.1 Baseline Features

By treating the semantic argument classification task as a classification problem using machine learning, one of the most important steps in building an accurate classifier is the choice of appropriate features. Most features used in prior research, such as [Gildea and Jurafsky, 2002; Pradhan *et al.*, 2005; Bejan *et al.*, 2004], can be categorized into three types:

- *Sentence level features*, such as predicate, voice, and predicate subcategorization.
- *Argument-specific features*, such as phrase type, head word, content word, head word's part of speech, and named entity class of the argument.
- *Argument-predicate relational features*, such as an argument's position with respect to the predicate, and its syntactic path to the predicate.

The above features attempt to capture the syntactic environment of the semantic argument currently being classified. They are entirely determined by the underlying shallow parse or full parse of the sentence. They carry no information about the semantic arguments that have already been identified or classified. As such, the classification of each semantic argument is done independently from other semantic arguments. It assumes that the semantic arguments of the same predicate do not influence each other.

We use the baseline features in [Pradhan *et al.*, 2005] as our baseline. These features are explained in Table 1.

2.2 Semantic Context Features

For a semantic argument, its semantic context features are defined as the features of its neighboring semantic arguments. The combination of the features of the current semantic argument and its neighboring semantic arguments encodes the interdependence among the semantic arguments.

To illustrate, the parse tree in Figure 1 is annotated with the semantic arguments of the predicate "added". These semantic arguments are $ARG1$, $ARG2$, $ARG4$, and $ARGM\text{-}ADV$.

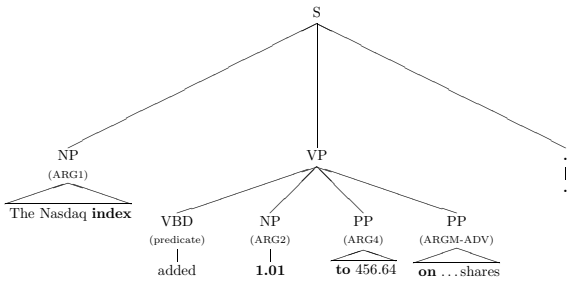

Figure 1: Semantically labeled parse tree, from dev set 00

Pr	Vo	Sc	Pt	Hw	Pa	Po	Ar
add	active	VP:VBD_NP_PP_PP	NP	index	NP∧S_VP_VBD	L	ARG1
add	active	VP:VBD_NP_PP_PP	NP	1.01	NP∧VP_VBD	R	ARG2
add	active	VP:VBD_NP_PP_PP	PP	to	PP∧VP_VBD	R	ARG4
add	active	VP:VBD_NP_PP_PP	PP	on	PP∧VP_VBD	R	ARGM-ADV

Table 2: Semantic context features based on Figure 1

Each row of Table 2 contains the baseline features (as defined in Table 1) extracted for each semantic argument.

In Figure 1, suppose the semantic argument identification module has correctly identified the constituents "The Nasdaq index", "1.01", "to 456.64", and "on . . . shares" as semantic arguments of the predicate "added". Assume further that the semantic argument "The Nasdaq index" has been assigned the semantic role $ARG1$. Now consider the task of assigning a semantic role to the semantic argument "1.01". Without using semantic context features, the baseline features used for the semantic argument "1.01" appear in row 2 of Table 2.

If we augment with the semantic context features of all neighboring arguments, then the features Pt, Hw, Pa, and Po of the neighboring arguments $ARG1$, $ARG4$, and $ARGM\text{-}ADV$ are added as additional features. Also, since "The Nasdaq index" has been assigned a semantic role, its argument class $ARG1$ is also used as an additional semantic context feature Ar.

2.3 Various Ways of Incorporating Semantic Context Features

There are many possible subsets of the semantic context features in Table 2 that one can choose to add to the baseline features of the current semantic argument being classified. In addition, since the semantic role labels of arguments that are already classified (like $ARG1$ in the previous example) are available as additional context features, the order in which we process the semantic arguments during semantic argument classification can affect the accuracy of the semantic role labels assigned. A good classifier should incorporate an informative set of semantic context features, as well as adopt an argument ordering that helps to give the best overall accuracy for semantic argument classification.

We use context feature acronyms with subscripts to denote particular types of context features at specific locations relative to the current argument being classified. For example, Hw_1 refers to the head word feature of the argument imme-

Feature	Description	Examples
Pt_i	the syntactic category of the ith context semantic argument	Pt_{-1}=NP ; Pt_1=PP
Hw_i	the head word of the ith context semantic argument	Hw_{-1}=index ; Hw_1=to
Pa_i	the path of the ith context semantic argument	Pa_{-1}=NP∧S_VP_VBD ; Pa_1=PP∧VP_VBD
Po_i	the position of the ith context semantic argument	Po_{-1}=L ; Po_1=R
Ar_i	the semantic role of the ith previous semantic argument	Ar_{-1}=ARG1

Table 3: Examples of semantic context features. The current semantic argument being classified is "1.01" as shown in Figure 1.

diately to the right of the current argument being classified. More examples are given in Table 3. We also use set notation $\{-i..i\}$ to denote the set of context features with subscript index $j \in \{-i, \ldots, i\}$. For example, $Hw_{\{-1..1\}}$ denotes the context features Hw_{-1} and Hw_1.

Adding All Types of Context Features

We could choose to add all types of context features within a certain window size. To illustrate with the example in Figure 1, suppose the current semantic argument being classified is "1.01". If we add all types of context features from the set $\{-1..1\}$ (i.e., within a window size of one), then the additional context features added are those in Table 4 that are darkly shaded. The baseline features for the current argument "1.01" is lightly shaded in Table 4.

Pr	Vo	Sc	Pt	Hw	Pa	Po	Ar
add	active	VP:VBD_NP_PP_PP	NP	index	NP∧S_VP_VBD	L	ARG1
add	active	VP:VBD_NP_PP_PP	NP	1.01	NP∧VP_VBD	R	*
add	active	VP:VBD_NP_PP_PP	PP	to	PP∧VP_VBD	R	
add	active	VP:VBD_NP_PP_PP	PP	on	PP∧VP_VBD	R	

Table 4: Adding all types of context features from the set $\{-1..1\}$

Adding a Single Type of Context Features

Alternatively, we could add only a specific type of context features within a certain window size, since it may be the case that certain types of context features are more effective than others. To illustrate again with the same example. If we add the head word features $Hw_{\{-2..2\}}$ (i.e., within a window size of two), then the additional context features added are those in Table 5 that are darkly shaded. The baseline features for the current argument "1.01" is lightly shaded in Table 5.

Different Argument Ordering

So far we have implicitly assumed that the semantic arguments are processed in a *linear ordering*, i.e., according to the natural textual order in which the semantic arguments appear in a sentence. In PropBank, the semantic arguments of a single predicate in a sentence do not overlap, so this ordering

Pr	Vo	Sc	Pt	Hw	Pa	Po	Ar
				nil			
add	active	VP:VBD_NP_PP_PP	NP	index	NP∧S_VP_VBD	L	ARG1
add	active	VP:VBD_NP_PP_PP	NP	1.01	NP∧VP_VBD	R	*
add	active	VP:VBD_NP_PP_PP	PP	to	PP∧VP_VBD	R	
add	active	VP:VBD_NP_PP_PP	PP	on	PP∧VP_VBD	R	
				nil			

Table 5: Adding one type of context features $Hw_{\{-2..2\}}$

is well-defined. Using a linear ordering, the semantic arguments in Figure 1 are processed in the order $ARG1$, $ARG2$, $ARG4$, and $ARGM\text{-}ADV$.

Experiments reported in [Lim *et al.*, 2004] indicate that semantic arguments spanned by the parent of the predicate node (the immediate clause) can be more accurately classified. Hence, if we process these semantic arguments first, their semantic role labels Ar_i may be more accurately determined and so may help in the subsequent classification of the remaining semantic arguments. Inspired by [Lim *et al.*, 2004], we view a parse tree as containing subtrees of increasing size, with each subtree rooted at each successive ancestor node of the predicate node. In *subtree ordering*, the semantic arguments spanned by a smaller subtree are processed first before the arguments of the next larger subtree. Using a subtree ordering, the semantic arguments in Figure 1 are processed in the order $ARG2$, $ARG4$, $ARGM\text{-}ADV$, and $ARG1$.

3 An Illustrating Example of the Utility of Semantic Context Features

In this section, we shall motivate the utility of semantic context features by considering the example in Figure 1. The predicate "add" has three rolesets defined in PropBank, as shown in Table 6. Each roleset corresponds to a different argument structure. Depending on the particular roleset that a specific instance of "add" belongs to, different argument classes are assigned to its semantic arguments.

Roleset	Arguments
add.01	ARG0(speaker), ARG1(utterance)
add.02	ARG0(adder), ARG1(thing being added), ARG2(thing being added to)
add.03	ARG1(logical subject, patient, thing rising / gaining / being added to), ARG2(amount risen), ARG4(end point), ARGM-LOC(medium)

Table 6: The rolesets of "add", as defined in PropBank

The roleset for "add" in our example is "add.03", where the first argument to the left of "added", "The Nasdaq index", is the "logical subject, patient, thing rising / gaining / being added to". This contrasts with the second roleset "add.02", such as "added" in the sentence "Judge Curry added an additional $55 million to the commission's calculations.", which is an illustrating example used in PropBank's description of the roleset "add.02".

During classification of the semantic arguments "1.01" and "to 456.64" in this example, using only the baseline fea-

tures of an individual semantic argument results in "1.01" being misclassified as $ARG1$ and "to 456.64" being misclassified as $ARG2$, when their correct semantic argument classes should be $ARG2$ and $ARG4$ respectively. Since the two rolesets "add.02" and "add.03" have similar argument structures (add something to something), and since "add.02" occurs more frequently than "add.03", the semantic argument classification module has assigned the semantic argument classes of the roleset "add.02" to "1.01" and "to 456.64", which is incorrect.

However, our experiments in section 4 show that a classifier augmented with the context features $Hw_{\{-2..2\}}$ assigns the correct semantic argument classes to "1.01" and "to 456.64". This is because in the sentence "index added something to something", that "index" is the head word of the first argument gives a strong clue that the correct roleset is "add.03", since "index" cannot fill the adder (agent) role of "add.02". In fact, occurrence counts from PropBank section 02-21 indicate that of the three occurrences of "index" as the head word of the first argument of predicate "add", all three appear in the roleset "add.03". This example illustrates the importance of neighboring semantic arguments in the context of the current semantic argument in determining its correct semantic argument class.

4 Experimental Results

Our semantic argument classification module uses support vector learning and is implemented with yamcha[3] and tinysvm[4] [Kudo and Matsumoto, 2001]. Polynomial kernel with degree 2 is used in a one-vs-all classifier. The cost per unit violation of the margin $C = 1$, and the tolerance of the termination criterion $e = 0.001$.

The training, development, and test data sections follow the conventional split of Section 02-21, 00, and 23 of PropBank release I respectively. In Section 4.1 to Section 4.4, we present the accuracy scores when evaluated on development section 00. In Section 4.5, we present the accuracy scores on test section 23.

The semantic argument classification accuracy when evaluated on section 00 using baseline features is 88.16%, assuming that semantic argument identification is done correctly (i.e., "gold argument identification"). In Section 4.1 to Section 4.4, accuracy scores that show a statistically significant improvement (χ^2 test with $p = 0.05$) over the baseline score of 88.16% are marked by "*". The highest score of each row in the score tables is boldfaced.

4.1 Results of Linear Ordering

Results Using All Types of Context Features

Table 7 contains accuracy scores after augmenting baseline features with all types of semantic context features. Ar feature is only present within context feature sets that contain neighboring arguments to the left of the current argument. We observe that the accuracy scores have a significant improvement from the baseline if context features on both sides of the current argument are used together. Context features to the left of the current argument are not effective when used alone.

	Accuracy of increasing context window size				
feature	$\{-1..1\}$	$\{-2..2\}$	$\{-3..3\}$	$\{-4..4\}$	$\{-5..5\}$
all	89.27*	**89.32***	88.91	88.52	88.20
feature	$\{-1..0\}$	$\{-2..0\}$	$\{-3..0\}$	$\{-4..0\}$	$\{-5..0\}$
all	**87.88**	87.41	87.14	86.84	86.70
feature	$\{0..1\}$	$\{0..2\}$	$\{0..3\}$	$\{0..4\}$	$\{0..5\}$
all	88.74	**88.82**	88.48	88.32	88.08

Table 7: Accuracy based on adding all types of semantic context features, with increasing window size, assuming correct argument identification and linear ordering of processing arguments

Results Using a Single Type of Context Features

Accuracy scores of semantic argument classification after augmenting baseline features with a single type of context features are shown in Table 8. The best improvement results from the use of head word features Hw. The use of features Ar reduces accuracy. This is likely due to semantic argument classes not being accurately determined, and errors committed in a position in linear ordering might affect classification accuracy at subsequent positions. However, a better argument ordering may improve the effectiveness of Ar.

4.2 Results of Subtree Ordering

We repeat the experiments of the last subsection, but this time with the use of subtree ordering. The accuracy scores are given in Table 9 and Table 10. The most prominent difference from the results of linear ordering is the better accuracy scores with the use of the Ar features, as shown in Table 10 compared with Table 8. The improvement observed is consistent with the findings of [Lim *et al.*, 2004], that the argument class of the semantic arguments spanned by the parent of the predicate node can be more accurately determined.

4.3 More Experiments with the Ar Feature

Unlike other semantic context features that are completely determined once argument identification is completed, the Ar feature is dynamically determined during argument classification. It may be possible to improve the overall accuracy of

	Accuracy of increasing context window size				
feature	$\{-1..1\}$	$\{-2..2\}$	$\{-3..3\}$	$\{-4..4\}$	$\{-5..5\}$
Pt	89.01*	**89.06***	88.89	88.81	88.55
Hw	89.49*	**89.82***	89.56*	89.42*	89.32*
Pa	**89.17***	89.04*	88.89	88.66	88.55
Po	88.63	**88.74**	88.58	88.54	88.24
feature	$\{-1..0\}$	$\{-2..0\}$	$\{-3..0\}$	$\{-4..0\}$	$\{-5..0\}$
Ar	**87.98**	87.72	87.73	87.78	87.75

Table 8: Accuracy based on adding a single type of semantic context features, with increasing window size, assuming correct argument identification and linear ordering of processing arguments

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

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

	Accuracy of increasing context window size				
feature	$\{-1..1\}$	$\{-2..2\}$	$\{-3..3\}$	$\{-4..4\}$	$\{-5..5\}$
all	**88.99***	88.96*	88.62	88.43	88.33
feature	$\{-1..0\}$	$\{-2..0\}$	$\{-3..0\}$	$\{-4..0\}$	$\{-5..0\}$
all	**88.92**	88.83	88.87	88.66	88.40
feature	$\{0..1\}$	$\{0..2\}$	$\{0..3\}$	$\{0..4\}$	$\{0..5\}$
all	**88.41**	88.38	88.08	87.80	87.63

Table 9: Accuracy based on adding all types of semantic context features, with increasing window size, assuming correct argument identification and subtree ordering of processing arguments

	Accuracy of increasing context window size				
feature	$\{-1..1\}$	$\{-2..2\}$	$\{-3..3\}$	$\{-4..4\}$	$\{-5..5\}$
Pt	**89.03***	88.98*	88.83	88.77	88.46
Hw	89.29*	**89.60***	89.29*	89.20*	89.08*
Pa	89.25*	**89.25***	89.14*	88.95	88.74
Po	88.77	**88.96***	88.72	88.37	88.21
feature	$\{-1..0\}$	$\{-2..0\}$	$\{-3..0\}$	$\{-4..0\}$	$\{-5..0\}$
Ar	88.85	88.84	**88.90**	88.87	88.83

Table 10: Accuracy based on adding a single type of semantic context features, with increasing window size, assuming correct argument identification and subtree ordering of processing arguments

assigning argument classes to all the semantic arguments of a predicate if we use a beam search algorithm to keep the best k combinations of argument classes assigned to all semantic arguments processed so far.

Ar Feature with Gold Semantic Argument Class

To determine the maximum accuracy improvement possible with the use of the Ar feature, we assume that the argument classes of all previous semantic arguments have been accurately determined and used as semantic context features. The experimental results are presented in Table 11, titled "linear gold" and "subtree gold" respectively for linear and subtree ordering.

Ar Feature with Beam Search

Tinysvm's output scores are converted to confidence scores between 0 and 1, by applying the sigmoid function $y = \frac{1}{1+e^{-x}}$. We implemented a beam search algorithm to find the overall best sequence of argument classes to be assigned to all semantic arguments. At each iteration, the beam search algorithm processes one semantic argument and determines the top 3 argument classes with the highest confidence scores for each semantic argument. It then finds and keeps the best k (k = 10 in our implementation) argument class sequences overall. Each argument class sequence is assigned a confidence score, which is the product of the confidence scores of all arguments in the sequence. Finally, the sequence of argument classes for all semantic arguments with the highest score is chosen. Experimental results are presented under "linear beam" and "subtree beam" in Table 11, and show improvements after using beam search with Ar feature for subtree ordering.

	Accuracy of increasing context window size				
Ar with	$\{-1..0\}$	$\{-2..0\}$	$\{-3..0\}$	$\{-4..0\}$	$\{-5..0\}$
linear	**87.98**	87.72	87.73	87.78	87.75
linear beam	**88.83**	88.46	88.64	88.66	88.67
linear gold	89.37*	89.54*	89.59*	**89.60***	89.58*
subtree	88.85	88.84	**88.90**	88.87	88.83
subtree beam	89.13*	**89.20***	89.18*	89.19*	89.20*
subtree gold	89.87*	89.84*	90.09*	90.07*	**90.11***

Table 11: More experiments with the Ar feature, using beam search and gold semantic argument class history

4.4 Combining Multiple Semantic Context Features

The best accuracy score we have obtained so far is 89.82%, given by $Hw_{-2..2}$ in Table 8. This score is a statistically significant improvement over the baseline score of 88.16% obtained with only baseline features. We want to further leverage on the other types of semantic context features. Error analysis of experiments in Table 8 and 10 on development section 00 shows that classifiers using different semantic context features make different classification mistakes. Thus we would like to explore the combination of multiple semantic context features. This provides a basis for combining multiple semantic context features. Here we more carefully select the context features than simply using all available features as in Table 7 and 9.

A Single Classifier with Multiple Context Features

The best context features from each row of Table 8, $Pt_{\{-2..2\}}$, $Hw_{\{-2..2\}}$, $Pa_{\{-1..1\}}$, $Po_{\{-2..2\}}$, and $Ar_{\{-1..0\}}$ are combined to form a new classifier based on linear ordering. Similarly $Pt_{\{-1..1\}}$, $Hw_{\{-2..2\}}$, $Pa_{\{-2..2\}}$, $Po_{\{-2..2\}}$, and $Ar\{-3..0\}$ from Table 10 are combined to form a new classifier based on subtree ordering. Evaluation on development section 00 gives accuracy scores of 89.54% and 89.19%, for linear and subtree ordering, respectively. The lack of accuracy improvement over that of $Hw_{-2..2}$ shows that such a simple combination might accumulate the errors introduced by each additional context feature.

Voting with Multiple Classifiers

Instead of building a new classifier with multiple semantic context features, one can combine the classification results of multiple classifiers and hope to achieve better accuracy through consensus and mutual error correction. The voting process combines k different classifiers each belonging to one row in Table 8 or Table 10. Currently only classifiers based on the same ordering are combined. Classifiers are chosen from different context feature types, but can be of the same context feature window size. For a particular semantic argument A, each candidate argument class will accumulate its confidence score assigned by all the voting classifiers. The candidate argument class with the highest aggregate score will be output as the argument class for A. Exhaustive experiments are carried out with $k = 2, 3, 4, 5$. On the development section 00, the best voting classifier in linear ordering is $\{Pt_{\{-1..1\}}, Hw_{\{-4..4\}}, Pa_{\{-2..2\}}, Ar_{\{-1..0\}}\}$, with accuracy 90.32%. The best for subtree ordering is

Linear ordering		Subtree ordering	
Feature	Accuracy	Feature	Accuracy
$Pt_{\{-2..2\}}$	89.26*	$Pt_{\{-1..1\}}$	88.88
$Hw_{\{-2..2\}}$	89.92*	$Hw_{\{-2..2\}}$	89.66*
$Pa_{\{-1..1\}}$	89.20*	$Pa_{\{-2..2\}}$	89.34*
$Po_{\{-2..2\}}$	88.96	$Po_{\{-2..2\}}$	89.20*
$Ar_{\{-1..0\}}$	88.14	$Ar_{\{-3..0\}}$	88.88
$Ar^{beam}_{\{-1..0\}}$	88.80	$Ar^{beam}_{\{-2..0\}}$	89.13*
$All_{\{-2..2\}}$	89.61*	$All_{\{-1..1\}}$	89.33*
$Vote_{linear}$	90.15*	$Vote_{subtree}$	90.50*

Table 12: Semantic argument classification accuracy on test section 23. Baseline accuracy is 88.41%.

$\{Hw_{\{-2..2\}}, Pa_{\{-1..1\}}, Ar_{\{-4..0\}}\}$ with accuracy 90.62%. We denote the two voting classifiers as $Vote_{linear}$ and $Vote_{subtree}$ respectively.

Note that the accuracy score of 90.62% achieved by $Vote_{subtree}$ is a statistically significant improvement over the best score we have obtained without voting, 89.82% of $Hw_{\{-2..2\}}$ in Table 8.

4.5 Testing on Section 23

Based on the experiments on development section 00, we choose the best classifiers and apply them for semantic argument classification on test section 23. The baseline classification accuracy for section 23 is 88.41%, with the use of only baseline features. Subtree voting improves the accuracy to 90.50%, representing a relative error reduction of 18%. The detailed scores are in Table 12.

5 Related Work

To overcome the inadequate assumption of semantic argument independence during classification, [Punyakanok *et al.*, 2004] employed integer programming to impose global constraints in semantic role labeling. These global constraints can be viewed as an introduction of semantic context knowledge, e.g., no argument classes should be duplicated. However, these constraints are rule-based and may not always be applicable due to exceptions. Our proposed approach can be viewed as inducing the constraints via statistical learning.

The methods proposed in [Hacioglu *et al.*, 2004; Kouchnir, 2004; Park *et al.*, 2004] used features such as the phrase type and head word of neighboring syntactic chunks. Experiments in [Pradhan *et al.*, 2005; Fleischman *et al.*, 2003; Kwon *et al.*, 2004] used the semantic role labels of previous semantic arguments as dynamic context features and showed improved accuracy. This demonstrates the utility of limited semantic context features. The semantic context features proposed in this paper capture more extensively the interdependence among arguments.

Our proposed subtree ordering can be viewed as a generalization of the two-level incremental approach of [Lim *et al.*, 2004] in processing arguments, in which a sentence is divided into *immediate clause* and *upper clauses*.

6 Conclusion

In this paper, we successfully used additional features of the neighboring semantic arguments in determining the class of the current semantic argument. Our experimental results have shown significant improvement in the accuracy of semantic argument classification after exploiting argument interdependence. Argument classification accuracy on the standard Section 23 test set yields a relative error reduction of 18%.

Acknowledgements

We thank Taku Kudo and Sameer Pradhan for their help in answering our questions related to the work in this paper.

References

[Baker *et al.*, 1998] C. F. Baker, C. J. Fillmore, and J. B. Lowe. The Berkeley FrameNet Project. *Proceedings of COLING*, 1998.

[Bejan *et al.*, 2004] C. A. Bejan, A. Moschitti, P. Morarescu, G. Nicolae, and S. Harabagiu. Semantic Parsing Based on FrameNet. *Proceedings of SENSEVAL-3*, 2004.

[Fleischman *et al.*, 2003] M. Fleischman, N. Kwon, and E. Hovy. Maximum Entropy Models for FrameNet Classification. *Proceedings of EMNLP*, 2003.

[Gildea and Jurafsky, 2002] D. Gildea and D. Jurafsky. Automatic Labeling of Semantic Roles. *Computational Linguistics*, 2002.

[Hacioglu *et al.*, 2004] K. Hacioglu, S. Pradhan, W. Ward, J. H. Martin, and D. Jurafsky. Semantic Role Labeling by Tagging Syntactic Chunks. *Proceedings of CoNLL*, 2004.

[Kingsbury *et al.*, 2002] P. Kingsbury, M. Palmer, and M. Marcus. Adding Semantic annotation to the Penn Treebank. *Proceedings of HLT*, 2002.

[Kouchnir, 2004] B. Kouchnir. A Memory-based Approach for Semantic Role Labeling. *Proceedings of CoNLL*, 2004.

[Kudo and Matsumoto, 2001] T. Kudo and Y. Matsumoto. Chunking with Support Vector Machines. *Proceedings of NAACL*, 2001.

[Kwon *et al.*, 2004] N. Kwon, M. Fleischman, and E. Hovy. FrameNet-based Semantic Parsing using Maximum Entropy Models. *Proceedings of COLING*, 2004.

[Lim *et al.*, 2004] J.-H. Lim, Y.-S. Hwang, S.-Y. Park, and H.-C. Rim. Semantic Role Labeling using Maximum Entropy Model. *Proceedings of CoNLL*, 2004.

[Park *et al.*, 2004] K.-M. Park, Y.-S. Hwang, and H.-C. Rim. Two-Phase Semantic Role Labeling based on Support Vector Machines. *Proceedings of CoNLL*, 2004.

[Pradhan *et al.*, 2005] S. Pradhan, K. Hacioglu, V. Krugler, W. Ward, J. H. Martin, and D. Jurafsky. Support Vector Learning for Semantic Argument Classification. *To appear in Machine Learning*, 2005.

[Punyakanok *et al.*, 2004] V. Punyakanok, D. Roth, W.-T. Yih, and D. Zimak. Semantic Role Labeling via Integer Linear Programming Inference. *Proceedings of COLING*, 2004.