

# A Hybrid Convolution Tree Kernel for Semantic Role Labeling

**Wanxiang Che**  
Harbin Inst. of Tech.  
Harbin, China, 150001  
car@ir.hit.edu.cn

**Min Zhang**  
Inst. for Infocomm Research  
Singapore, 119613  
mzhang@i2r.a-star.edu.sg

**Ting Liu, Sheng Li**  
Harbin Inst. of Tech.  
Harbin, China, 150001  
{tliu, ls}@ir.hit.edu.cn

## Abstract

A hybrid convolution tree kernel is proposed in this paper to effectively model syntactic structures for semantic role labeling (SRL). The hybrid kernel consists of two individual convolution kernels: a Path kernel, which captures predicate-argument link features, and a Constituent Structure kernel, which captures the syntactic structure features of arguments. Evaluation on the datasets of CoNLL-2005 SRL shared task shows that the novel hybrid convolution tree kernel outperforms the previous tree kernels. We also combine our new hybrid tree kernel based method with the standard rich flat feature based method. The experimental results show that the combinational method can get better performance than each of them individually.

## 1 Introduction

In the last few years there has been increasing interest in Semantic Role Labeling (SRL). It is currently a well defined task with a substantial body of work and comparative evaluation. Given a sentence, the task consists of analyzing the propositions expressed by some target verbs and some constituents of the sentence. In particular, for each target verb (predicate) all the constituents in the sentence which fill a semantic role (argument) of the verb have to be recognized.

Figure 1 shows an example of a semantic role labeling annotation in PropBank (Palmer et al., 2005). The PropBank defines 6 main arguments, Arg0 is the *Agent*, Arg1 is *Patient*, etc. ArgM may indicate adjunct arguments, such as *Locative*, *Temporal*.

Many researchers (Gildea and Jurafsky, 2002; Pradhan et al., 2005a) use feature-based methods

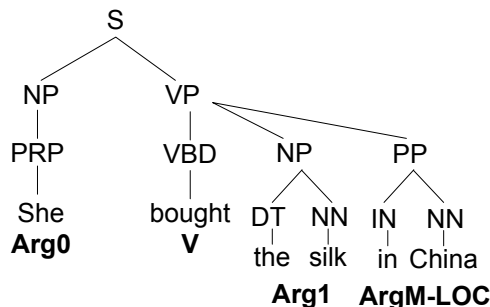


Figure 1: Semantic role labeling in a phrase structure syntactic tree representation

for argument identification and classification in building SRL systems and participating in evaluations, such as Senseval-3<sup>1</sup>, CoNLL-2004 and 2005 shared tasks: SRL (Carreras and Màrquez, 2004; Carreras and Màrquez, 2005), where a flat feature vector is usually used to represent a predicate-argument structure. However, it's hard for this kind of representation method to explicitly describe syntactic structure information by a vector of flat features. As an alternative, convolution tree kernel methods (Collins and Duffy, 2001) provide an elegant kernel-based solution to implicitly explore tree structure features by directly computing the similarity between two trees. In addition, some machine learning algorithms with dual form, such as Perceptron and Support Vector Machines (SVM) (Cristianini and Shawe-Taylor, 2000), which do not need know the exact presentation of objects and only need compute their kernel functions during the process of learning and prediction. They can be well used as learning algorithms in the kernel-based methods. They are named kernel machines.

In this paper, we decompose the Moschitti (2004)'s predicate-argument feature (PAF) kernel into a Path kernel and a Constituent Structure ker-

<sup>1</sup><http://www.cs.unt.edu/~rada/senseval/senseval3/>

nel, and then compose them into a hybrid convolution tree kernel. Our hybrid kernel method using Voted Perceptron kernel machine outperforms the PAF kernel in the development sets of CoNLL-2005 SRL shared task. In addition, the final composing kernel between hybrid convolution tree kernel and standard features’ polynomial kernel outperforms each of them individually.

The remainder of the paper is organized as follows: In Section 2 we review the previous work. In Section 3 we illustrate the state of the art feature-based method for SRL. Section 4 discusses our method. Section 5 shows the experimental results. We conclude our work in Section 6.

## 2 Related Work

Automatic semantic role labeling was first introduced by Gildea and Jurafsky (2002). They used a linear interpolation method and extract features from a parse tree to identify and classify the constituents in the FrameNet (Baker et al., 1998) with syntactic parsing results. Here, the basic features include Phrase Type, Parse Tree Path, Position. Most of the following works focused on feature engineering (Xue and Palmer, 2004; Jiang et al., 2005) and machine learning models (Nielsen and Pradhan, 2004; Pradhan et al., 2005a). Some other works paid much attention to the robust SRL (Pradhan et al., 2005b) and post inference (Punyakankok et al., 2004).

These feature-based methods are considered as the state of the art method for SRL and achieved much success. However, as we know, the standard flat features are less effective to model the syntactic structured information. It is sensitive to small changes of the syntactic structure features. This can give rise to a data sparseness problem and prevent the learning algorithms from generalizing unseen data well.

As an alternative to the standard feature-based methods, kernel-based methods have been proposed to implicitly explore features in a high-dimension space by directly calculating the similarity between two objects using kernel function. In particular, the kernel methods could be effective in reducing the burden of feature engineering for structured objects in NLP problems. This is because a kernel can measure the similarity between two discrete structured objects directly using the original representation of the objects instead of explicitly enumerating their features.

Many kernel functions have been proposed in machine learning community and have been applied to NLP study. In particular, Haussler (1999) and Watkins (1999) proposed the best-known convolution kernels for a discrete structure. In the context of convolution kernels, more and more kernels for restricted syntaxes or specific domains, such as string kernel for text categorization (Lodhi et al., 2002), tree kernel for syntactic parsing (Collins and Duffy, 2001), kernel for relation extraction (Zelenko et al., 2003; Culotta and Sorensen, 2004) are proposed and explored in NLP domain. Of special interest here, Moschitti (2004) proposed Predicate Argument Feature (PAF) kernel under the framework of convolution tree kernel for SRL. In this paper, we follow the same framework and design a novel hybrid convolution kernel for SRL.

## 3 Feature-based methods for SRL

Usually feature-based methods refer to the methods which use the flat features to represent instances. At present, most of the successful SRL systems use this method. Their features are usually extended from Gildea and Jurafsky (2002)’s work, which uses flat information derived from a parse tree. According to the literature, we select the Constituent, Predicate, and Predicate-Constituent related features shown in Table 1.

Feature	Description
Constituent related features	
Phrase Type	syntactic category of the constituent
Head Word	head word of the constituent
Last Word	last word of the constituent
First Word	first word of the constituent
Named Entity	named entity type of the constituent’s head word
POS	part of speech of the constituent
Previous Word	sequence previous word of the constituent
Next Word	sequence next word of the constituent
Predicate related features	
Predicate	predicate lemma
Voice	grammatical voice of the predicate, either active or passive
SubCat	Sub-category of the predicate’s parent node
Predicate POS	part of speech of the predicate
Suffix	suffix of the predicate
Predicate-Constituent related features	
Path	parse tree path from the predicate to the constituent
Position	the relative position of the constituent and the predicate, before or after
Path Length	the nodes number on the parse tree path
Partial Path	some part on the parse tree path
Clause Layers	the clause layers from the constituent to the predicate

Table 1: Standard flat features

However, to find relevant features is, as usual, a complex task. In addition, according to the description of the standard features, we can see that the syntactic features, such as Path, Path Length, bulk large among all features. On the other hand, the previous researches (Gildea and Palmer, 2002; Punyakankok et al., 2005) have also recognized the

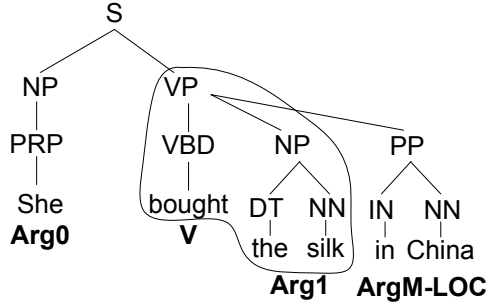


Figure 2: Predicate Argument Feature space

necessity of syntactic parsing for semantic role labeling. However, the standard flat features cannot model the syntactic information well. A predicate-argument pair has two different Path features even if their paths differ only for a node in the parse tree. This data sparseness problem prevents the learning algorithms from generalizing unseen data well. In order to address this problem, one method is to list all sub-structures of the parse tree. However, both space complexity and time complexity are too high for the algorithm to be realized.

#### 4 Hybrid Convolution Tree Kernels for SRL

In this section, we introduce the previous kernel method for SRL in Subsection 4.1, discuss our method in Subsection 4.2 and compare our method with previous work in Subsection 4.3.

##### 4.1 Convolution Tree Kernels for SRL

Moschitti (2004) proposed to apply convolution tree kernels (Collins and Duffy, 2001) to SRL. He selected portions of syntactic parse trees, which include salient sub-structures of predicate-arguments, to define convolution kernels for the task of predicate argument classification. This portions selection method of syntactic parse trees is named as predicate-arguments feature (PAF) kernel. Figure 2 illustrates the PAF kernel feature space of the predicate *buy* and the argument **Arg1** in the circled sub-structure.

The kind of convolution tree kernel is similar to Collins and Duffy (2001)’s tree kernel except the sub-structure selection strategy. Moschitti (2004) only selected the relative portion between a predicate and an argument.

Given a tree portion instance defined above, we design a convolution tree kernel in a way similar to the parse tree kernel (Collins and Duffy, 2001).

Firstly, a parse tree  $T$  can be represented by a vector of integer counts of each sub-tree type (regardless of its ancestors):

$$\Phi(T) = (\# \text{ of sub-trees of type } 1, \dots, \\ \# \text{ of sub-trees of type } i, \dots, \\ \# \text{ of sub-trees of type } n)$$

This results in a very high dimension since the number of different subtrees is exponential to the tree’s size. Thus it is computationally infeasible to use the feature vector  $\Phi(T)$  directly. To solve this problem, we introduce the tree kernel function which is able to calculate the dot product between the above high-dimension vectors efficiently. The kernel function is defined as following:

$$K(T_1, T_2) = \langle \Phi(T_1), \Phi(T_2) \rangle = \sum_i \phi_i(T_1) \phi_i(T_2) \\ = \sum_{n_1 \in N_1} \sum_{n_2 \in N_2} \sum_i I_i(n_1) * I_i(n_2)$$

where  $N_1$  and  $N_2$  are the sets of all nodes in trees  $T_1$  and  $T_2$ , respectively, and  $I_i(n)$  is the indicator function whose value is 1 if and only if there is a sub-tree of type  $i$  rooted at node  $n$  and 0 otherwise. Collins and Duffy (2001) show that  $K(T_1, T_2)$  is an instance of convolution kernels over tree structures, which can be computed in  $O(|N_1| \times |N_2|)$  by the following recursive definitions (Let  $\Delta(n_1, n_2) = \sum_i I_i(n_1) * I_i(n_2)$ ):

- (1) if the children of  $n_1$  and  $n_2$  are different then  $\Delta(n_1, n_2) = 0$ ;
- (2) else if their children are the same and they are leaves, then  $\Delta(n_1, n_2) = \mu$ ;
- (3) else  $\Delta(n_1, n_2) = \mu \prod_{j=1}^{nc(n_1)} (1 + \Delta(ch(n_1, j), ch(n_2, j)))$

where  $nc(n_1)$  is the number of the children of  $n_1$ ,  $ch(n, j)$  is the  $j^{th}$  child of node  $n$  and  $\mu (0 < \mu < 1)$  is the decay factor in order to make the kernel value less variable with respect to the tree sizes.

##### 4.2 Hybrid Convolution Tree Kernels

In the PAF kernel, the feature spaces are considered as an integral portion which includes a predicate and one of its arguments. We note that the PAF feature consists of two kinds of features: one is the so-called parse tree *Path* feature and another one is the so-called *Constituent Structure* feature. These two kinds of feature spaces represent different information. The Path feature describes the

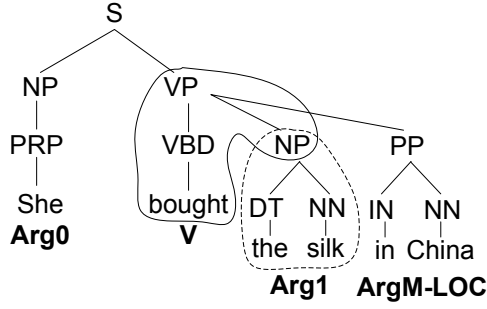


Figure 3: Path and Constituent Structure feature spaces

linking information between a predicate and its arguments while the Constituent Structure feature captures the syntactic structure information of an argument. We believe that it is more reasonable to capture the two different kinds of features separately since they contribute to SRL in different feature spaces and it is better to give different weights to fuse them. Therefore, we propose two convolution kernels to capture the two features, respectively and combine them into one hybrid convolution kernel for SRL. Figure 3 is an example to illustrate the two feature spaces, where the Path feature space is circled by solid curves and the Constituent Structure feature spaces is circled by dotted curves. We name them Path kernel and Constituent Structure kernel respectively.

Figure 4 illustrates an example of the distinction between the PAF kernel and our kernel. In the PAF kernel, the tree structures are equal when considering constitutes *NP* and *PRP*, as shown in Figure 4(a). However, the two constituents play different roles in the sentence and should not be looked as equal. Figure 4(b) shows the computing example with our kernel. During computing the hybrid convolution tree kernel, the NP-PRP substructure is not computed. Therefore, the two trees are distinguished correctly.

On the other hand, the constituent structure feature space reserves the most part in the traditional PAF feature space usually. Then the Constituent Structure kernel plays the main role in PAF kernel computation, as shown in Figure 5. Here, *believes* is a predicate and **A1** is a long sub-sentence. According to our experimental results in Section 5.2, we can see that the Constituent Structure kernel does not perform well. Affected by this, the PAF kernel cannot perform well, either. However, in our hybrid method, we can adjust the compromise

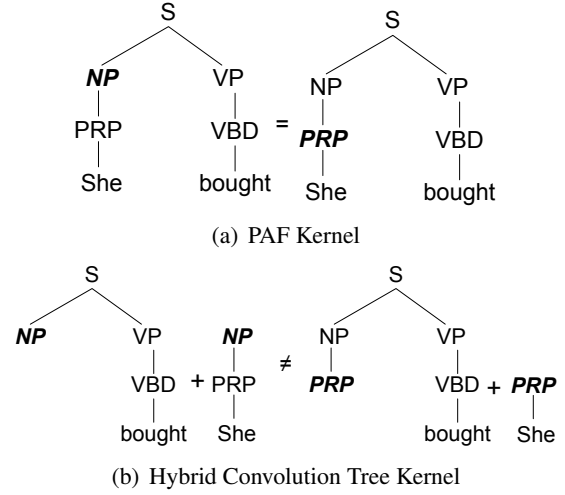


Figure 4: Comparison between PAF and Hybrid Convolution Tree Kernels

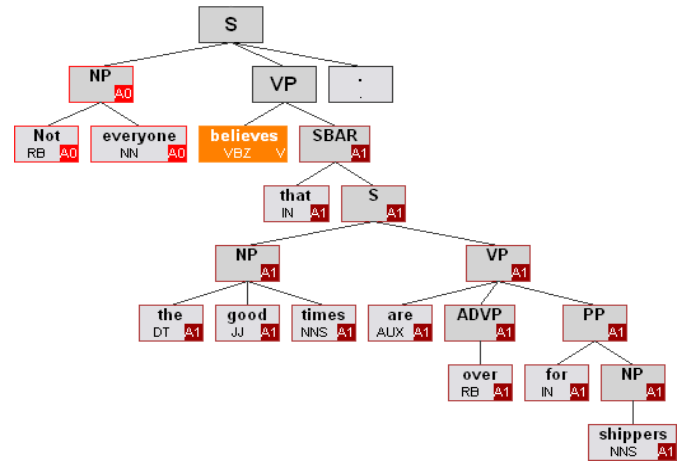


Figure 5: An example of Semantic Role Labeling

of the Path feature and the Constituent Structure feature by tuning their weights to get an optimal result.

Having defined two convolution tree kernels, the Path kernel  $K_{path}$  and the Constituent Structure kernel  $K_{cs}$ , we can define a new kernel to compose and extend the individual kernels. According to Joachims et al. (2001), the kernel function set is closed under linear combination. It means that the following  $K_{hybrid}$  is a valid kernel if  $K_{path}$  and  $K_{cs}$  are both valid.

$$K_{hybrid} = \lambda K_{path} + (1 - \lambda) K_{cs} \quad (1)$$

where  $0 \leq \lambda \leq 1$ .

According to the definitions of the Path and the Constituent Structure kernels, each kernel is explicit. They can be viewed as a matching of fea-

tures. Since the features are enumerable on the given data, the kernels are all valid. Therefore, the new kernel  $K_{hybrid}$  is valid. We name the new kernel hybrid convolution tree kernel,  $K_{hybrid}$ .

Since the size of a parse tree is not constant, we normalize  $K(T_1, T_2)$  by dividing it by  $\sqrt{K(T_1, T_1) \cdot K(T_2, T_2)}$

### 4.3 Comparison with Previous Work

It would be interesting to investigate the differences between our method and the feature-based methods. The basic difference between them lies in the instance representation (parse tree vs. feature vector) and the similarity calculation mechanism (kernel function vs. dot-product). The main difference between them is that they belong to different feature spaces. In the kernel methods, we implicitly represent a parse tree by a vector of integer counts of each sub-tree type. That is to say, we consider all the sub-tree types and their occurring frequencies. In this way, on the one hand, the predicate-argument related features, such as Path, Position, in the flat feature set are embedded in the Path feature space. Additionally, the Predicate, Predicate POS features are embedded in the Path feature space, too. The constituent related features, such as Phrase Type, Head Word, Last Word, and POS, are embedded in the Constituent Structure feature space. On the other hand, the other features in the flat feature set, such as Named Entity, Previous, and Next Word, Voice, SubCat, Suffix, are not contained in our hybrid convolution tree kernel. From the syntactic viewpoint, the tree representation in our feature space is more robust than the Parse Tree Path feature in the flat feature set since the Path feature is sensitive to small changes of the parse trees and it also does not maintain the hierarchical information of a parse tree.

It is also worth comparing our method with the previous kernels. Our method is similar to the Moschitti (2004)’s predicate-argument feature (PAF) kernel. However, we differentiate the Path feature and the Constituent Structure feature in our hybrid kernel in order to more effectively capture the syntactic structure information for SRL. In addition Moschitti (2004) only study the task of argument classification while in our experiment, we report the experimental results on both identification and classification.

## 5 Experiments and Discussion

The aim of our experiments is to verify the effectiveness of our hybrid convolution tree kernel and and its combination with the standard flat features.

### 5.1 Experimental Setting

#### 5.1.1 Corpus

We use the benchmark corpus provided by CoNLL-2005 SRL shared task (Carreras and Màrquez, 2005) provided corpus as our training, development, and test sets. The data consist of sections of the Wall Street Journal (WSJ) part of the Penn TreeBank (Marcus et al., 1993), with information on predicate-argument structures extracted from the PropBank corpus (Palmer et al., 2005). We followed the standard partition used in syntactic parsing: sections 02-21 for training, section 24 for development, and section 23 for test. In addition, the test set of the shared task includes three sections of the Brown corpus. Table 2 provides counts of sentences, tokens, annotated propositions, and arguments in the four data sets.

	Train	Devel	tWSJ	tBrown
Sentences	39,832	1,346	2,416	426
Tokens	950,028	32,853	56,684	7,159
Propositions	90,750	3,248	5,267	804
Arguments	239,858	8,346	14,077	2,177

Table 2: Counts on the data set

The preprocessing modules used in CONLL-2005 include an SVM based POS tagger (Giménez and Màrquez, 2003), Charniak (2000)’s full syntactic parser, and Chieu and Ng (2003)’s Named Entity recognizer.

#### 5.1.2 Evaluation

The system is evaluated with respect to *precision*, *recall*, and  $F_{\beta=1}$  of the predicted arguments. *Precision* ( $p$ ) is the proportion of arguments predicted by a system which are correct. *Recall* ( $r$ ) is the proportion of correct arguments which are predicted by a system.  $F_{\beta=1}$  computes the harmonic mean of *precision* and *recall*, which is the final measure to evaluate the performances of systems. It is formulated as:  $F_{\beta=1} = 2pr/(p + r)$ . *srl-eval.pl*<sup>2</sup> is the official program of the CoNLL-2005 SRL shared task to evaluate a system performance.

<sup>2</sup><http://www.lsi.upc.edu/~srlconll/srl-eval.pl>

### 5.1.3 SRL Strategies

We use constituents as the labeling units to form the labeled arguments. In order to speed up the learning process, we use a four-stage learning architecture:

**Stage 1:** To save time, we use a pruning stage (Xue and Palmer, 2004) to filter out the constituents that are clearly not semantic arguments to the predicate.

**Stage 2:** We then identify the candidates derived from Stage 1 as either arguments or non-arguments.

**Stage 3:** A multi-category classifier is used to classify the constituents that are labeled as arguments in Stage 2 into one of the argument classes plus NULL.

**Stage 4:** A rule-based post-processing stage (Liu et al., 2005) is used to handle some unmatched arguments with constituents, such as AM-MOD, AM-NEG.

### 5.1.4 Classifier

We use the Voted Perceptron (Freund and Schapire, 1998) algorithm as the kernel machine. The performance of the Voted Perceptron is close to, but not as good as, the performance of SVM on the same problem, while saving computation time and programming effort significantly. SVM is too slow to finish our experiments for tuning parameters.

The Voted Perceptron is a binary classifier. In order to handle multi-classification problems, we adopt the *one vs. others* strategy and select the one with the largest margin as the final output. The training parameters are chosen using development data. After 5 iteration numbers, the best performance is achieved. In addition, Moschitti (2004)’s Tree Kernel Tool is used to compute the tree kernel function.

## 5.2 Experimental Results

In order to speed up the training process, in the following experiments, we **ONLY** use WSJ sections 02-05 as training data. The same as Moschitti (2004), we also set the  $\mu = 0.4$  in the computation of convolution tree kernels.

In order to study the impact of  $\lambda$  in hybrid convolution tree kernel in Eq. 1, we only use the hybrid kernel between  $K_{path}$  and  $K_{cs}$ . The perfor-

mance curve on development set changing with  $\lambda$  is shown in Figure 6.

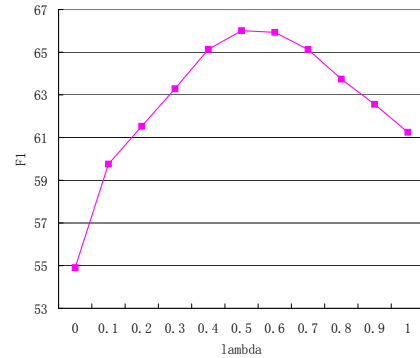


Figure 6: The performance curve changing with  $\lambda$

The performance curve shows that when  $\lambda = 0.5$ , the hybrid convolution tree kernel gets the best performance. Either the Path kernel ( $\lambda = 1$ ,  $F_{\beta=1} = 61.26$ ) or the Constituent Structure kernel ( $\lambda = 0$ ,  $F_{\beta=1} = 54.91$ ) cannot perform better than the hybrid one. It suggests that the two individual kernels are complementary to each other. In addition, the Path kernel performs much better than the Constituent Structure kernel. It indicates that the predicate-constituent related features are more effective than the constituent features for SRL.

Table 3 compares the performance comparison among our Hybrid convolution tree kernel, Moschitti (2004)’s PAF kernel, standard flat features with Linear kernels, and Poly kernel ( $d = 2$ ). We can see that our hybrid convolution tree kernel outperforms the PAF kernel. It empirically demonstrates that the weight linear combination in our hybrid kernel is more effective than PAF kernel for SRL.

However, our hybrid kernel still performs worse than the standard feature based system. This is simple because our kernel only use the syntactic structure information while the feature-based method use a large number of hand-craft diverse features, from word, POS, syntax and semantics, NER, etc. The standard features with polynomial kernel gets the best performance. The reason is that the arbitrary binary combination among features implicated by the polynomial kernel is useful to SRL. We believe that combining the two methods can perform better.

In order to make full use of the syntactic information and the standard flat features, we present a composite kernel between hybrid kernel ( $K_{hybrid}$ ) and standard features with polynomial

	Hybrid	PAF	Linear	Poly
Devel	66.01	64.38	68.71	<b>70.25</b>

Table 3: Performance ( $F_{\beta=1}$ ) comparison among various kernels

kernel ( $K_{poly}$ ):

$$K_{comp} = \gamma K_{hybrid} + (1 - \gamma) K_{poly} \quad (2)$$

where  $0 \leq \gamma \leq 1$ .

The performance curve changing with  $\gamma$  in Eq. 2 on development set is shown in Figure 7.

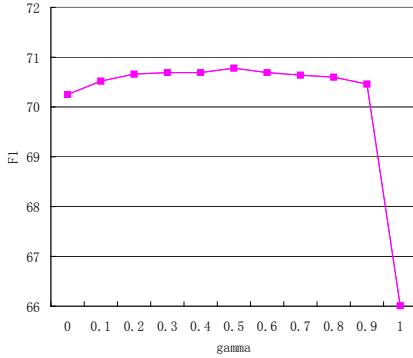


Figure 7: The performance curve changing with  $\gamma$

We can see that when  $\gamma = 0.5$ , the system achieves the best performance and  $F_{\beta=1} = 70.78$ . It's statistically significant improvement ( $\chi^2$  test with  $p = 0.1$ ) than only using the standard features with the polynomial kernel ( $\gamma = 0$ ,  $F_{\beta=1} = 70.25$ ) and much higher than only using the hybrid convolution tree kernel ( $\gamma = 1$ ,  $F_{\beta=1} = 66.01$ ). The main reason is that the convolution tree kernel can represent more general syntactic features than standard flat features, and the standard flat features include the features that the convolution tree kernel cannot represent, such as Voice, Sub-Cat. The two kind features are complementary to each other.

Finally, we train the composite method using the above setting (Eq. 2 with when  $\gamma = 0.5$ ) on the entire training set. The final performance is shown in Table 4.

## 6 Conclusions and Future Work

In this paper we proposed the hybrid convolution kernel to model syntactic structure information for SRL. Different from the previous convolution tree kernel based methods, our contribution

	Precision	Recall	$F_{\beta=1}$
Development	80.71%	68.49%	74.10
Test WSJ	82.46%	70.65%	76.10
Test Brown	73.39%	57.01%	64.17

Test WSJ	Precision	Recall	$F_{\beta=1}$
Overall	82.46%	70.65%	76.10
A0	87.97%	82.49%	85.14
A1	80.51%	71.69%	75.84
A2	75.79%	52.16%	61.79
A3	80.85%	43.93%	56.93
A4	83.56%	59.80%	69.71
A5	100.00%	20.00%	33.33
AM-ADV	66.27%	43.87%	52.79
AM-CAU	68.89%	42.47%	52.54
AM-DIR	56.82%	29.41%	38.76
AM-DIS	79.02%	75.31%	77.12
AM-EXT	73.68%	43.75%	54.90
AM-LOC	72.83%	50.96%	59.97
AM-MNR	68.54%	42.44%	52.42
AM-MOD	98.52%	96.37%	97.43
AM-NEG	97.79%	96.09%	96.93
AM-PNC	49.32%	31.30%	38.30
AM-TMP	82.15%	68.17%	74.51
R-A0	86.28%	87.05%	86.67
R-A1	80.00%	74.36%	77.08
R-A2	100.00%	31.25%	47.62
R-AM-CAU	100.00%	50.00%	66.67
R-AM-EXT	50.00%	100.00%	66.67
R-AM-LOC	92.31%	57.14%	70.59
R-AM-MNR	20.00%	16.67%	18.18
R-AM-TMP	68.75%	63.46%	66.00
V	98.65%	98.65%	98.65

Table 4: Overall results (top) and detailed results on the WSJ test (bottom).

is that we distinguish between the Path and the Constituent Structure feature spaces. Evaluation on the datasets of CoNLL-2005 SRL shared task, shows that our novel hybrid convolution tree kernel outperforms the PAF kernel method. Although the hybrid kernel base method is not as good as the standard rich flat feature based methods, it can improve the state of the art feature-based methods by implicating the more generalizing syntactic information.

Kernel-based methods provide a good framework to use some features which are difficult to model in the standard flat feature based methods. For example the semantic similarity of words can be used in kernels well. We can use general purpose corpus to create clusters of similar words or use available resources like WordNet. We can also use the hybrid kernel method into other tasks, such as relation extraction in the future.

## Acknowledgements

The authors would like to thank the reviewers for their helpful comments and Shiqi Zhao, Yanyan Zhao for their suggestions and useful discussions. This work was supported by National Natural Science Foundation of China (NSFC) via grant 60435020, 60575042, and 60503072.

## References

- Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In *Proceedings of the ACL-Coling-1998*, pages 86–90.
- Xavier Carreras and Lluís Màrquez. 2004. Introduction to the CoNLL-2004 shared task: Semantic role labeling. In *Proceedings of CoNLL-2004*, pages 89–97.
- Xavier Carreras and Lluís Màrquez. 2005. Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *Proceedings of CoNLL-2005*, pages 152–164.
- Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of NAACL-2000*.
- Hai Leong Chieu and Hwee Tou Ng. 2003. Named entity recognition with a maximum entropy approach. In *Proceedings of CoNLL-2003*, pages 160–163.
- Michael Collins and Nigel Duffy. 2001. Convolution kernels for natural language. In *Proceedings of NIPS-2001*.
- Nello Cristianini and John Shawe-Taylor. 2000. *An Introduction to Support Vector Machines*. Cambridge University Press, Cambridge University.
- Aron Culotta and Jeffrey Sorensen. 2004. Dependency tree kernels for relation extraction. In *Proceedings of ACL-2004*, pages 423–429.
- Yoav Freund and Robert E. Schapire. 1998. Large margin classification using the perceptron algorithm. In *Computational Learning Theory*, pages 209–217.
- Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.
- Daniel Gildea and Martha Palmer. 2002. The necessity of parsing for predicate argument recognition. In *Proceedings of ACL-2002*, pages 239–246.
- Jesús Giménez and Lluís Màrquez. 2003. Fast and accurate part-of-speech tagging: The svm approach revisited. In *Proceedings of RANLP-2003*.
- David Haussler. 1999. Convolution kernels on discrete structures. Technical Report UCSC-CRL-99-10, July.
- Zheng Ping Jiang, Jia Li, and Hwee Tou Ng. 2005. Semantic argument classification exploiting argument interdependence. In *Proceedings of IJCAI-2005*.
- Thorsten Joachims, Nello Cristianini, and John Shawe-Taylor. 2001. Composite kernels for hypertext categorisation. In *Proceedings of ICML-2001*, pages 250–257.
- Ting Liu, Wanxiang Che, Sheng Li, Yuxuan Hu, and Huaijun Liu. 2005. Semantic role labeling system using maximum entropy classifier. In *Proceedings of CoNLL-2005*, pages 189–192.
- Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. 2002. Text classification using string kernels. *Journal of Machine Learning Research*, 2:419–444.
- Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: the penn treebank. *Computational Linguistics*, 19(2):313–330.
- Alessandro Moschitti. 2004. A study on convolution kernels for shallow statistic parsing. In *Proceedings of ACL-2004*, pages 335–342.
- Rodney D. Nielsen and Sameer Pradhan. 2004. Mixing weak learners in semantic parsing. In *Proceedings of EMNLP-2004*.
- Martha Palmer, Dan Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1).
- Sameer Pradhan, Kadri Hacioglu, Valeri Krugler, Wayne Ward, James H. Martin, and Daniel Jurafsky. 2005a. Support vector learning for semantic argument classification. *Machine Learning Journal*.
- Sameer Pradhan, Wayne Ward, Kadri Hacioglu, James Martin, and Daniel Jurafsky. 2005b. Semantic role labeling using different syntactic views. In *Proceedings of ACL-2005*, pages 581–588.
- Vasin Punyakanok, Dan Roth, Wen-tau Yih, and Dav Zimak. 2004. Semantic role labeling via integer linear programming inference. In *Proceedings of Coling-2004*, pages 1346–1352.
- Vasin Punyakanok, Dan Roth, and Wen tau Yih. 2005. The necessity of syntactic parsing for semantic role labeling. In *Proceedings of IJCAI-2005*, pages 1117–1123.
- Chris Watkins. 1999. Dynamic alignment kernels. Technical Report CSD-TR-98-11, Jan.
- Nianwen Xue and Martha Palmer. 2004. Calibrating features for semantic role labeling. In *Proceedings of EMNLP 2004*.
- Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *Journal of Machine Learning Research*, 3:1083–1106.