Using a Hybrid Convolution Tree Kernel for Semantic Role Labeling

WANXIANG CHE
Harbin Institute of Technology
MIN ZHANG and AI TI AW
Institute for Infocomm Research
CHEW LIM TAN
National University of Singapore
and
TING LIU and SHENG LI
Harbin Institute of Technology

As a kind of Shallow Semantic Parsing, Semantic Role Labeling (SRL) is gaining more attention as it benefits a wide range of natural language processing applications. Given a sentence, the task of SRL is to recognize semantic arguments (roles) for each predicate (target verb or noun). Feature-based methods have achieved much success in SRL and are regarded as the state-of-the-art methods for SRL. However, these methods are less effective in modeling structured features. As an extension of feature-based methods, kernel-based methods are able to capture structured features more efficiently in a much higher dimension. Application of kernel methods to SRL has been achieved by selecting the tree portion of a predicate and one of its arguments as feature space, which is named as predicate-argument feature (PAF) kernel. The PAF kernel captures the syntactic tree structure features using convolution tree kernel, however, it does not distinguish between the path structure and the constituent structure. In this article, a hybrid convolution tree kernel is proposed to model different linguistic objects. The hybrid convolution tree kernel consists of two individual convolution tree kernels. They are a Path kernel, which captures predicate-argument link features, and a Constituent Structure kernel, which captures the syntactic structure features of arguments. Evaluations on the data sets of the CoNLL-2005 SRL shared task and the Chinese PropBank (CPB) show that our proposed hybrid convolution tree kernel statistically significantly

The majority of this work was done when W. Che was an intern at the Institute for Infocomm Research.
Some preliminary results have been reported at Coling/ACL 2006 [Che et al. 2006] and this article is an extended version of the Coling/ACL 2006 article.
Authors' addresses: W. Che, School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China, 150001; email: car@ir.hit.edu.cn; M. Zhang, email: mzhang@i2r.a-star.edu.sg.

outperforms the previous tree kernels. Moreover, in order to maximize the system performance, we present a composite kernel through combining our hybrid convolution tree kernel method with a feature-based method extended by the polynomial kernel. The experimental results show that the composite kernel achieves better performance than each of the individual methods and outperforms the best reported system on the CoNLL-2005 corpus when only one syntactic parser is used and on the CPB corpus when automated syntactic parse results and correct syntactic parse results are used respectively.

Categories and Subject Descriptors: I.2.7 [**Artificial Intelligence**]: Natural Language Processing—*Semantic role labeling*; I.5.1 [**Pattern Recognition**]: Models—*kernel*; I.5.4 [**Pattern Recognition**]: Applications—*Text processing*

General Terms: Documentation, Languages

Additional Key Words and Phrases: Semantic role labeling, hybrid convolution tree kernel

ACM Reference Format:
Che, W., Zhang, M., Aw, A. T., Tan, C. L., Liu, T., and Li, S. 2008. Using a hybrid convolution tree kernel for semantic role labeling. ACM Trans. Asian Lang. Inform. Process. Article 13 (November 2008), 23 pages. DOI = 10.1145/1450295.1450298. http://doi.acm.org/10.1145/1450295.1450298.

1. INTRODUCTION

In recent years, there has been increasing interest in Shallow Semantic Parsing. It is becoming an important component in many kinds of deep natural language processing applications, such as question answering [Narayanan and Harbabagiu 2004; Moschitti et al. 2006; Shen and Lapata 2007], information extraction [Surdeanu et al. 2003], and coreference resolution [Ponzetto and Strube 2006]. As a particular case of shallow semantic parsing, Semantic Role Labeling (SRL) is currently a well-defined task with a substantial amount of work and comparative evaluation. Given a sentence, the task consists of analyzing the propositions expressed by some target verbs or nouns and some constituents of the sentence. In particular, all the constituents in the sentence which fulfill a semantic argument (role) for each predicate (target verb or noun) have to be recognized. Figure 1 shows an example of SRL annotation result in the English PropBank [Palmer et al. 2005]. It defines six core arguments (Arg0∼5), where Arg0 is the Agent, Arg1 is Patient, etc. ArgMs indicate adjunct arguments, such as ArgM-LOC (Locative), ArgM-TMP (Temporal). Chinese PropBank (CPB) [Xue and Kulick 2003] has a similar structure to the English PropBank. Both of them are widely-used benchmark data for SRL. We will introduce them in detail in the next section.

Generally, semantic role identification and classification are regarded as two key steps in SRL. Semantic role identification involves identifying the semantic and nonsemantic argument among all constituents in a sentence while semantic role classification involves classifying each identified semantic argument into a specific semantic role. Gildea and Jurafsky [2002] are the first to use a linear interpolation method and extracted features from a parse tree to identify and classify the constituents in the FrameNet Baker et al. [1998] with syntactic parsing results. Most of the following work focused on feature engineering [Pradhan et al. 2005; Xue and Palmer 2004; Jiang et al. 2005] and machine learning models [Nielsen and Pradhan 2004; Pradhan et al. 2005]. Some other work paid more attention to the robust SRL [Pradhan et al. 2005]

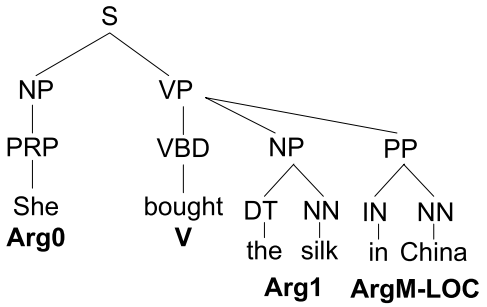


Fig. 1. An English SRL example in a phrase structure syntactic tree representation.

and post inference [Punyakanok et al. 2004]. Besides English, the SRL tasks on other languages, such as Chinese [Sun and Jurafsky 2004; Xue and Palmer 2005], were given more attention recently.

All the above work, including most systems participating in various shared tasks, such as the Senseval-3[1], the CoNLL-2004 and 2005 SRL shared tasks [Carreras and Màrquez 2004, 2005], used the state-of-the-art, feature-based methods for argument identification and classification. Feature-based methods usually use a flat feature vector to represent a learning object and are known to be hard in describing the syntactic structure information explicitly. As an alternative to the standard feature-based methods, kernel-based methods have been proposed to implicitly explore features in a high-dimensional space by directly calculating the similarity between two feature vectors, or even between two objects using a kernel function (Subsection 4.1 gives the formal definition of the kernel function). Moreover, there are some machine learning algorithms with dual form, such as Perceptron and support vector machines (SVM) [Cristianini and Shawe-Taylor 2000], which do not require the exact presentation of objects to compute their kernel functions during learning and prediction. Such algorithms can be used as learning algorithms in the kernel-based methods. They are named as kernel machines.

Many kernel functions proposed in the machine learning community have been applied to natural language processing tasks. In particular, Haussler [1999] and Watkins [1999] proposed the best-known convolution kernels for a discrete structure. Under the framework of convolution kernels, increasingly more kernels for restricted syntax or specific domains are proposed and explored, 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; Zhang et al. 2006b], and kernel for question answering [Moschitti et al. 2006]. Particularly, Moschitti [2004] and Moschitti et al. [forthcoming] proposed a Predicate Argument Feature (PAF) kernel under the framework of convolution tree kernel for SRL. It is considered as the first work using kernel-based methods for SRL. The PAF regards the minimum subtree comprising a predicate and a constituent

[1]http://www.cs.unt.edu/~rada/senseval/senseval3/

structure as the feature space. However, this point of view cannot distinguish among different kinds of features well, that is, the path feature and the constituent structure feature.

For the reasons described above, we propose a hybrid convolution tree kernel for SRL. We first decompose the PAF kernel into a Path kernel and a Constituent Structure kernel, and then combine them into a hybrid convolution tree kernel. That means the new kernel captures the Path feature and the Constituent Structure feature separately. This can model the structure features more effectively. Experiments on the test sets of the CoNLL-2005 SRL shared task and the Chinese PropBank show that our hybrid kernel method outperforms the PAF kernel significantly. In addition, in order to get the best performance, a composite kernel by combining our hybrid convolution tree kernel and a feature-based method extended by the polynomial kernel is presented. Experimental results show that the composite kernel outperforms each of the individual kernels. It also outperforms the best reported CoNLL-2005 shared task system which uses only one syntactic parser and the best reported system on the CPB corpus which uses gold parse trees.

The remaining of this article is organized as follows: In Section 2, we introduce the SRL corpora used in our experiments. In Section 3, we illustrate the state-of-the-art, feature-based methods for SRL while Section 4 introduces our method. The experimental results and discussion are shown in Section 5. Finally, our work is concluded in Section 6.

2. SEMANTIC ROLE LABELING CORPORA

The PropBank [Palmer et al. 2005] is a popular corpus for SRL in English. The latest version, English PropBank I, can be obtained from LDC (LDC2004T14)[2]. Correspondingly, the Chinese PropBank (CPB) [Xue and Kulick 2003] is a Chinese corpus for SRL.

In the English PropBank I, predicate-argument relations are annotated for verbs in the *Wall Street Journal* (WSJ) section of the Penn Treebank II [Marcus et al. 1993]. The arguments of a predicate contain six core arguments (Arg0$\sim$5) and 14 adjunctive arguments (ArgM-$*$). Usually, Arg0 denotes the *Agent* of a target verb; Arg1 is the Patient, and so on. ArgMs include ArgM-LOC (*Locative*), ArgM-TMP (*Temporal*), and so on. PropBank I was constructed by assigning semantic arguments to constituents of the hand-corrected Penn Treebank parse trees. So sometimes the parse trees can have *trace* nodes which refer to other nodes in the trees. Such nodes do not have any words associated with them but are also marked as arguments. For example, the trace node "NONE" is labeled as "Arg1" in this labeling result: "[The new plant $_{Arg1}$], located [-NONE- $_{Arg1}$] [in Chinchon $_{ArgM-LOC}$], ...". However these trace nodes cannot be reproduced by most automatic syntactic parsers.

Therefore, the CoNLL-2005 SRL shared task corpus [Carreras and Màrquez 2005], which is a snapshot of the PropBank with some corrections, is used in our experiments. The syntactic parse trees produced by automatic syntactic

[2]http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2004T14

Table I. The Statistical Information of the CoNLL-2005 SRL Shared Task Data Set and Chinese PropBank (CPB)

	CoNLL-2005				
	Train	Devel	Test (*WSJ*)	Test (Brown)	CPB
Sentences	39,832	1,346	2,416	426	10,367
Tokens	950,028	32,853	56,684	7,159	269,129
Propositions	90,750	3,248	5,267	804	36,849
Arguments	239,858	8,346	14,077	2,177	104,007

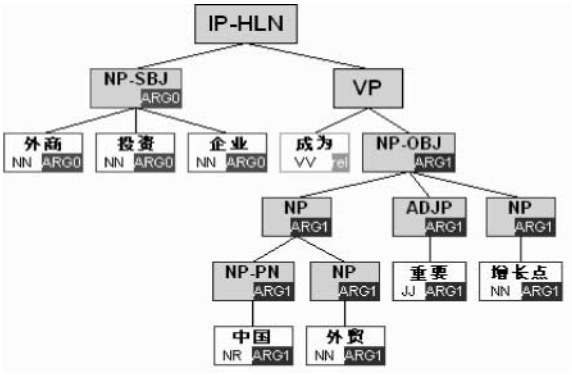


Fig. 2. A Chinese PropBank (CPB) example for sentence: "外商投资企业成为中国外贸重要增长点" (Foreign investment enterprises become an important growth entity for the Chinese foreign trade). Please note that we tag all of the nodes in the constituent of an argument. The same denote is applied to other figures.

parsers are used in the CoNLL-2005 corpus, including Charniak parser [Charniak 2000] and Collins parser [Collins 1999]. In addition, the preprocessing modules include an SVM based POS tagger [Gimenez and Màrquez 2003], and Chieu and Ng's [2003] Named Entity recognizer. At the same time, the CoNLL-2005 corpus annotates discontinuous and coreferential arguments explicitly with some simple rules. The first part of a discontinuous argument is labeled as it is, while the second part is labeled with a prefix "C-" appended to it (e.g., [The company $_{A0}$] [gained $_{V}$] [shareholder approval $_{A1}$] [Thursday $_{AM-TMP}$] [to restructure in a bid $_{C-A1}$].). All coreferential arguments are labeled with a prefix "R-" appended to them (e.g. [Every problem $_{A0}$] [that $_{R-A0}$] has [hobbled $_{V}$] [the program $_{A1}$] ....). In order to test the robustness of the systems, besides the *WSJ* corpus, a cross-corpora evaluation is performed using a fresh test set from the Brown corpus (ck01-03). It is provided by the PropBank team.

The standard partition of the CoNLL-2005 corpus is as follows: sections 02-21 for training, section 24 for development, and section 23 for testing. Some corpus statistics are listed in Table I.

The Chinese PropBank (CPB) [Xue and Kulick 2003] is based on the Penn Chinese TreeBank [Xue et al. 2005], which is a Chinese corpus annotated with syntactic structures. It is created by adding the semantic roles to the appropriate constituents of the syntactic tree. Figure 2 illustrates an example in the CPB. We note that the SRL annotation scheme of the CPB is similar to the English PropBank, that is, they use the same semantic role (arguments

Table II. Standard Flat Feature Set

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
Subcategory	Subcategory 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

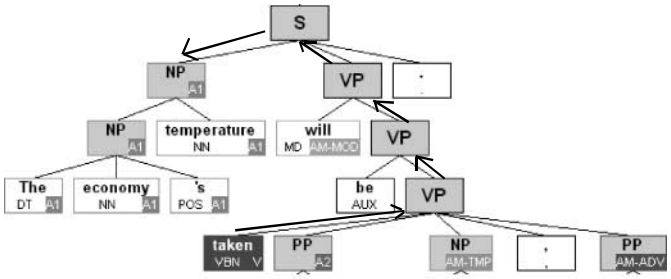
and adjuncts) set. Additionally, the syntactic structure of the Penn Chinese TreeBank is also similar to that of the Penn English TreeBank.

3. FEATURE-BASED METHODS FOR SRL

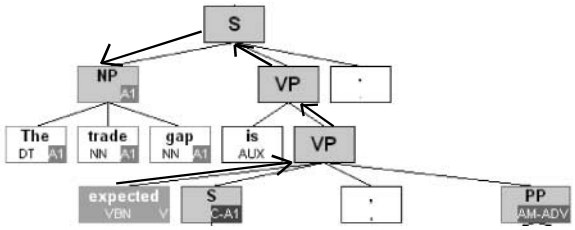
Feature-based methods refer to the standard methods which use a flat feature vector to represent an object. At present, most of the successful SRL systems use these methods. Their features are usually extended from Gildea and Jurafsky [2002]'s work, which used flat information derived from a parse tree. According to the literature, Gildea and Jurafsky [2002]; Pradhan et al. [2005] used the Constituent, Predicate, and Predicate-Constituent related features listed in Table II.

To find a useful feature set is usually a nontrivial task. In addition, earlier research [Gildea and Palmer 2002; Punyakanok et al. 2005] recognized the necessity and importance of syntactic parsing for SRL. Therefore, it is critical to effectively utilize the syntactic structure features in SRL. Unfortunately, feature-based methods are less effective in this respect. One of the reasons is that the standard feature-based methods intensify the data sparseness problem for syntactic structure features in SRL. This is because they are sensitive to small changes in the structures [Moschitti 2004]. For example, in Figure 3, the Path features between predicate and Arg1 in 3(a) and 3(b) are "VBN↑VP↑VP↑VP↑S↓NP" and "VBN↑VP↑VP↑S↓NP" respectively. Although their arguments are the same and their Path features differ only in one additional "VP↑", they have distinct Path features. This data sparseness problem prevents the learning algorithms from generalizing unseen data well.

To overcome this problem, Pradhan et al. [2005] tried generalizing the Path feature with some intuitive heuristics. However, the heuristic generalizing



(a) Path from predicate *take* to Arg1



(b) Path from predicate *expect* to Arg1

Fig. 3. Comparison between two Path features.

method is too restrictive to give a better coverage and is difficult to be used on other structure features. Another intuition is to represent a structure using all of its substructures. However, this generates a large amount of features which grow exponentially with respect to the size of the tree. In the next section, we will introduce our solution.

4. HYBRID CONVOLUTION TREE KERNELS FOR SRL

In this section, we first introduce the principle of kernel methods in Subsection 4.1 and traditional convolution tree kernels for SRL in Subsection 4.2. Then, we present our proposed hybrid convolution tree kernel for SRL in Subsection 4.3. Finally, we discuss the related work in Subsection 4.4.

4.1 Kernel-Based Methods

Kernel methods [Vapnik 1998; Shawe-Taylor and Cristianini 2004] are an attractive alternative to feature-based methods. The kernel methods retain the original representation of objects and use the object only via a *kernel function* (a special kind of similarity function) computed on a pair of objects. A kernel function K is a binary function over the object space X. That is $K : X \times X \rightarrow [0, \infty]$ maps a pair of objects $x, y \in X$ to their similarity score $K(x, y)$. The kernel (or similarity) function is required to be *symmetric*[3] and *positive-semidefinite*[4].

[3]A binary function $K(\cdot, \cdot)$ is symmetric (over X), if $\forall x, y \in X$, $K(x, y) = K(y, x)$.
[4]A binary function $K(\cdot, \cdot)$ is positive-semidefinite, if $\forall x_1, x_2, \ldots, x_n \in X$ the $n \times n$ matrix $(K(x_i, x_j))_{ij}$ is positive-semidefinite.

It can be shown that any kernel function implicitly calculates the dot-product of feature vectors of objects in high-dimensional feature spaces. That is, there exist features corresponding to a mapping function $\Phi(\cdot) = (\phi_1(\cdot), \phi_2(\cdot), \ldots), \phi_i : X \rightarrow R$, such that $K(\mathbf{x}, \mathbf{y}) = \langle \Phi(\mathbf{x}) \cdot \Phi(\mathbf{y}) \rangle$. Here $\langle \mathbf{a} \cdot \mathbf{b} \rangle$ denotes the dot product of vectors $\mathbf{a}$ and $\mathbf{b}$.

There are a number of learning algorithms that can operate using only the dot product (kernel function) of instances. We call them *kernel machines*. For instance, the support vector machine (SVM) is a learning algorithm that not only allows for kernel function, but also provides a rigorous rationale for resisting over-fitting [Vapnik 1998].

We note that, from the learning system design perspective, the kernel methods shift the focus from the problem of feature selection to the problem of kernel construction. Since a kernel is the only domain specific component of a kernel learning system, it is critical to design a kernel that adequately encapsulates all information necessary for learning. Next, we will show a special kind of kernel for some NLP tasks.

4.2 Convolution Tree Kernels for SRL

The convolution tree kernel counts the number of common subtrees (substructures) as the syntactic similarity between two parse trees. In the vector representation of a parse tree, a tree T can be represented by a vector of integer counts of each subtree type (regardless of its ancestors):

$$\begin{aligned}\Phi(T) &= (\phi_1(T), \phi_2(T), \ldots, \phi_n(T)) \\ &= (\#\text{ of sub-trees of type 1}, \\ &\quad \#\text{ of sub-trees of type 2}, \\ &\quad \ldots, \\ &\quad \#\text{ of sub-trees of type } n)\end{aligned}$$

This generates a very high-dimensional feature space 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, Collins and Duffy [2001] expanded Haussler [1999] and Watkins [1999]'s convolution kernel by developing a convolution tree kernel function which is able to calculate the dot product between the above high-dimensional vectors efficiently. The kernel function is defined as follows:

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

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 subtree of type i rooted at node n and 0 otherwise. Collins and Duffy [2001] showed that $K(T_1, T_2)$ is an instance of convolution kernels over tree structures, which

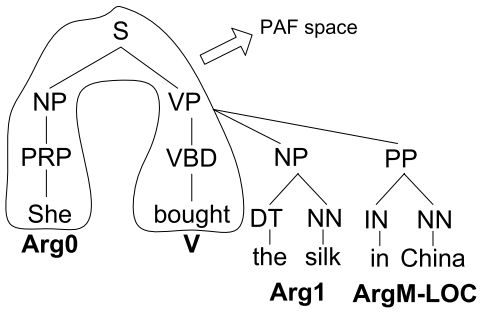


Fig. 4. Predicate Argument Feature space.

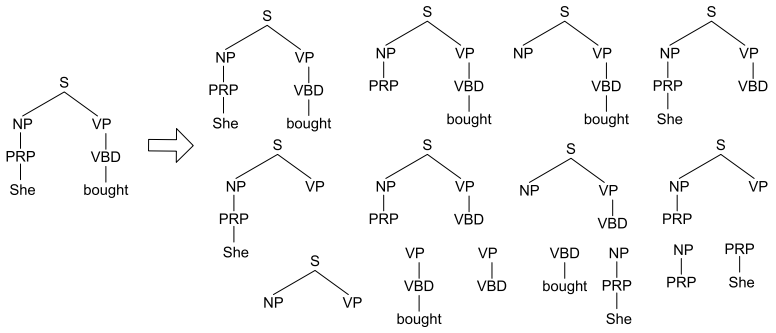


Fig. 5. All 15 subtrees extended from a PAF space.

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 production rules at 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 leave nodes, 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's size.

Moschitti [2004] proposed to apply the convolution tree kernels to SRL. He selected portions of syntactic parse trees, which include salient substructures of predicate-arguments as the predicate-arguments feature (PAF) space, and defined the convolution kernel on the PAF space. Figure 4 illustrates the PAF kernel feature space of the predicate *buy* and the argument Arg0 in the enclosed substructure. Figure 5 lists all the 15 subtree features in the PAF feature space. Besides the structure features, the PAF kernel covers many of the previous flat features, such as Predicate, Words, POSs. Moschitti [2004] further showed that the PAF kernel performs well on the semantic role classification subtask in SRL. By nature, the PAF kernel is similar to Collins and

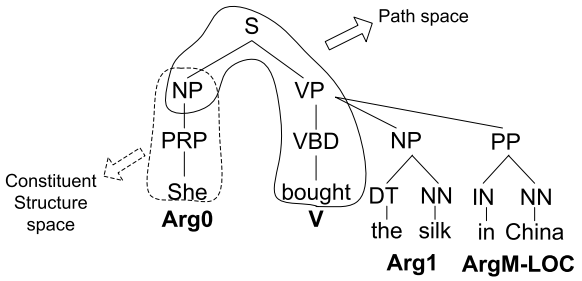


Fig. 6. Path and Constituent Structure feature spaces.

Duffy [2001]'s tree kernel except for the substructure selection strategy. More precisely, Moschitti [2004] only selected the relative portion between a predicate and an argument and defined the tree kernel over the selected portion.

4.3 Hybrid Convolution Tree Kernel for SRL

We note that the PAF feature space consists of two kinds of features namely the parse tree *Path* feature and the *Constituent Structure* feature. These two kinds of feature spaces represent different information. The Path feature captures the information between a predicate and its arguments while the Constituent Structure feature captures the syntactic structure information of an argument. It would be more reasonable to capture these two different kinds of features separately since they contribute to SRL in different ways. Then we can easily fuse them by a linear combination using different weights. Based on the above consideration, we propose two convolution kernels to capture the two features separately, and combine them into one hybrid convolution kernel for SRL. Figure 6 illustrates the two feature spaces, where the Path feature space is enclosed by a solid curve and the Constituent Structure feature space is enclosed by a dotted curve. We name them Path kernel and Constituent Structure kernel, respectively. Formally, the Path kernel is the tree kernel covering the smallest substructure which includes one predicate with the root node of a constituent subtree. The Constituent Structure kernel is the tree kernel covering a constituent.

Having defined the two convolution tree kernels, namely, the Path kernel K_{path} and the Constituent Structure kernel K_{cs}, we now define a new kernel to combine the two 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.

$$K_{hybrid}(T_1, T_2) = \lambda K_{path}(T_1, T_2) + (1 - \lambda)K_{cs}(T_1, T_2) \tag{1}$$

where $0 \leq \lambda \leq 1$, T_1 and T_2 are two syntactic trees.

Since the size of a parse tree is not constant, we normalize the $K_{path}(T_1, T_2)$ and the $K_{cs}(T_1, T_2)$ by dividing them by $\sqrt{K_{path}(T_1, T_1) \cdot K_{path}(T_2, T_2)}$ and $\sqrt{K_{cs}(T_1, T_1) \cdot K_{cs}(T_2, T_2)}$ respectively.

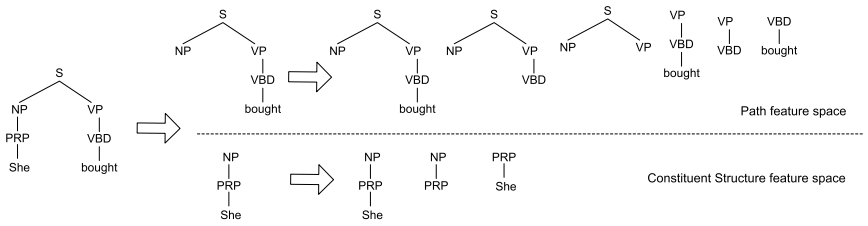
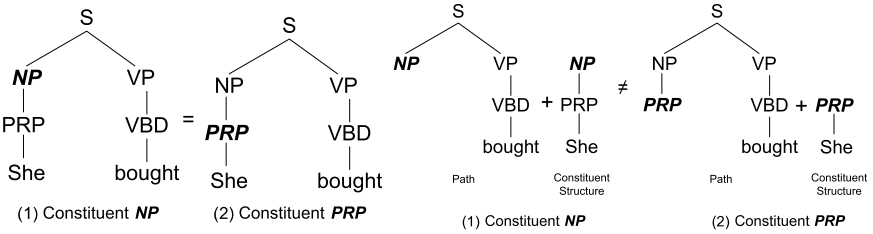


Fig. 7. The Path feature space comprises six subtrees and the Constituent Structure feature space comprises three subtrees.



(a) The PAF kernel (b) The hybrid convolution tree kernel

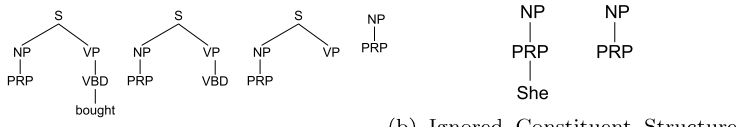
Fig. 8. Comparison between the PAF and the hybrid convolution tree kernels.

Unlike the feature space captured by the PAF kernel, the new feature space of the hybrid convolution tree kernel consists of two independent parts. Figure 7 illustrates the new feature space, where the Path feature space with six subtrees is listed above the dashed line, and the Constituent Structure feature space with three subtrees is listed below. Clearly, it is different from the PAF kernel's 15 subtrees listed in Figure 5.

Figure 8 illustrates the differences between the PAF kernel and our hybrid convolution tree kernel. In the PAF kernel, the tree structures are identical when considering constituents rooted at ***NP*** and ***PRP*** as arguments respectively, as shown in Figure 8(a). However, the two constituents play different roles in the sentence for predicate *buy* and should not be viewed as identical. Figure 8(b) shows the computing examples with the hybrid convolution tree kernel.

Figure 9 highlights the different subtrees between the two cases in Figure 8(b). Compared with the consideration of the case where the constituent rooted at ***NP*** as an argument [Figure 8(b)(1)], Figures 9(a) and 9(b) show the four additional features in the Path kernel feature space and the two ignored features in the Constituent Structure kernel feature space respectively, when considering the constituent rooted at ***PRP*** as an argument [Figure 8(b)(2)]. Therefore, the two trees could be distinguished correctly.

On the other hand, in most cases, the constituent structure feature space occupies the main part in the traditional PAF feature space. Statistics in the corpus of the CoNLL-2005 shared task shows that the size of a constituent structure is about twice the size of the path feature on average. Thus it plays a major role in the PAF kernel computation, as shown in Figure 10. Here, *go*



(a) Additional Path features (b) Ignored Constituent Structure features

Fig. 9. The different feature space when considering the constituent rooted at ***PRP*** as an argument comparing to considering the constituent rooted at ***NP*** as an argument when computing with the hybrid convolution tree kernel.

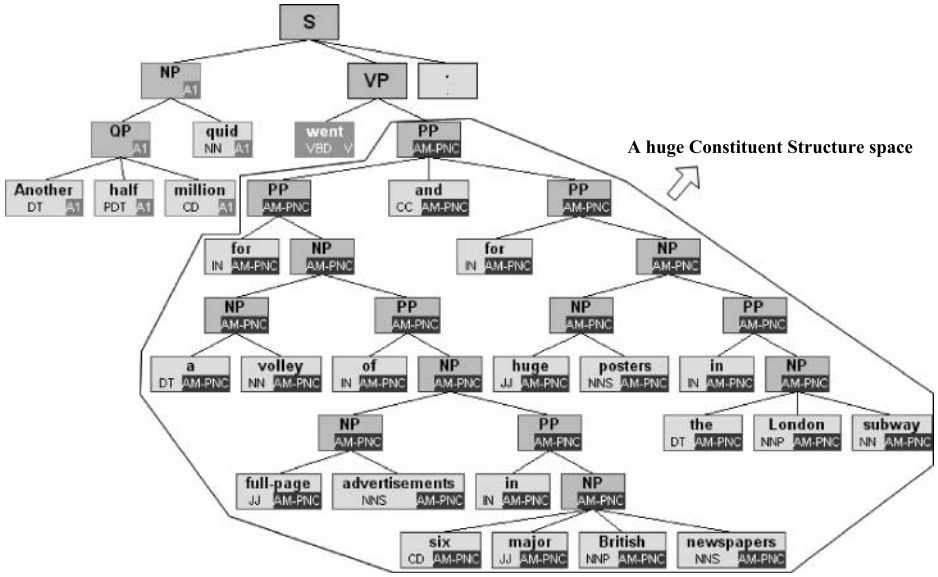


Fig. 10. An example of SRL showing that the imbalance between Path feature space (from predicate "went" to argument AM-PNC's root "PP") and Constituent feature space (AM-PNC substructure).

is a predicate and AM-PNC is a long subsentence. Our experimental results in Subsection 5.2 show that using the Constituent Structure kernel alone does not perform well. Since the Constituent Structure kernel dominates the PAF kernel score, the PAF kernel may not perform well, either. For example, if the final PAF kernel value is 0.9 (which is normalized by the size of the whole PAF structure), the constituent structure may contribute 0.6 whereas the path feature only contributes 0.3. Therefore, the contribution of the Path feature is not very significant in the final PAF kernel. In contrast, there is no such issue in our hybrid convolution tree kernel, since we have already normalized the Path kernel (K_{path}) and Constituent Structure kernel (K_{cs}) before combination. This can balance the contribution of the Constituent Structure feature and the Path feature, and therefore solve the problem found in the PAF kernel. We can also adjust the weights of the K_{path} and the K_{cs} to achieve an optimal

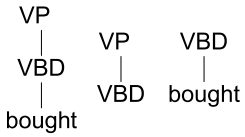


Fig. 11. For the two instances in Figure 8(a), there leaves only three common features for the MPAF kernel.

performance. Still for the example above, when we consider and normalize the two features separately, both of them may be contributing 0.45 (assuming that the combination weights are equal). Thus, the value of the Constituent Structure kernel is reduced relatively. The contribution of the Path feature is enhanced.

Finally, the combination of two identical subtrees, though they are coming from two different kinds of structures, confuses the Path and the Constituent Structure. For example, assuming there is a "VP→VBD" subtree in the Path of an instance and the same subtree occurs in the Constituent Structure of another instance, the two instances will have only one contribution in the PAF kernel. However, they should not be treated as equal as they belong to two different feature spaces. Our hybrid convolution tree kernel can overcome this problem well. In other words, our hybrid tree kernel uses different kernel functions to model different linguistic objects that describe different properties of the target linguistic phenomenon. From a machine learning perspective, we note that our hybrid kernel does not generate the substructures that tend to be less relevant for describing the target learning problem.

4.4 Related Work

Moschitti et al. [2006] also noted that the PAF kernel has the drawback for SRL, especially for argument identification. They provided an improved PAF (MPAF) kernel for SRL. In the MPAF kernel, the root node of a constituent is appended with "-B" symbol. Therefore, for the nodes "***NP***" and "***PRP***" in Figure 8(a), they become "***NP-B***" and "***PRP-B***" respectively. Thus, the new kernel can distinguish the boundary between the path and the constituent structures. Compared with our hybrid convolution tree kernel, there remain three common features for MPAF shown in Figure 11, which fail to capture the "S" included structures, such as "S→NP VP", which is a part of Path feature and an important information for SRL. On the other hand, although the MPAF method considers the boundary between the path and the constituent structures, it still treats them as an integral structure. Therefore, like the PAF kernel, it still depends mainly on the constituent related features, and is not flexible enough to consider the contribution of the Path and the Constituent Structure features separately. We will compare the MPAF kernel with our hybrid convolution tree kernel empirically.

Finally, to our knowledge, all the previous work on convolution tree kernel-based methods and their applications in natural language processing only used

one tree kernel on their problems, while we use two convolution tree kernels in a single application. Although Zhang et al. [2006a] compared various tree kernel spaces for relation extraction, they did not consider combining some of the convolution tree kernels together.

5. EXPERIMENTS AND DISCUSSION

The aim of our experiments is to verify the effectiveness of our hybrid convolution tree kernel and its combination with the feature-based method for SRL.

5.1 Experimental Setting

5.1.1 *Dataset.* As mentioned in Section 2, the CoNLL-2005 SRL shared task corpus is used as our English experimental dataset. We follow the standard partition using *WSJ* sections 02-21 for training, section 24 for development, and *WSJ* section 23 and the Brown corpus for testing.

As for the Chinese experiments, we use the Chinese PropBank 1.0[5], which consists of standoff annotation on all the 931 articles (chtb_001.fid to chtb_931.fid) of the Penn Chinese TreeBank 5.1[6]. To compare with Xue and Palmer's [2005] work, we follow their experimental setting, that is, dividing the CPB into the training data (661 files, chtb_100.fid to chtb_760.fid) and test data (other 99 files, chtb_001.fid to chtb_099.fid). In order to speed up the training process, we use the same parameters as the corresponding English methods tuned on the CoNLL-2005 development data. Therefore, no development data is used in the Chinese experiments. Note that the functional tags and traces are ignored for comparison with Xue and Palmer's [2005] work.

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 the systems. It is formulated as: $F_{\beta=1} = 2pr/(p + r)$. *srl-eval.pl*[7] is the official program of the CoNLL-2005 SRL shared task to evaluate a system performance.

5.1.3 *SRL Strategies.* We use constituents as the labeling units to form the labeled arguments. Because of the errors of the automatic syntactic parser, it is impossible for each argument to find its matching constituent in all parse trees. Statistics on the training set shows that 10.08% of the arguments have no matching constituents when we use the Charniak parser [Charniak 2000]. The number increases to 11.89% when we use the Collins parser [Collins 1999]. Therefore, we select the more accurate Charniak parser as the preprocessing module in our SRL system.

[5]http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2005T23
[6]http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2005T01U01
[7]http://www.lsi.upc.edu/∼srlconll/srl-eval.pl

In order to speed up the learning process, we use a four-stage learning architecture as follows:

Stage 1: To save time, we use a pruning stage [Xue and Palmer 2004, 2005] 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 using a binary classifier.

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.

Stage 4: A rule-based post-processing stage [Liu et al. 2005] is used for post inference. For the embedded arguments or the arguments with the same labels, we maintain the one with the maximum score.

5.1.4 *Classifier*. Support vector machines (SVM) [Vapnik 1998] are selected as our classifier Stages for 2 and 3. The SVM is a binary classifier. In order to handle multi-classification problem in Stage 3, we adopt the one vs. others strategy [Rifkin and Klautau 2004] and select the label with the largest score output by a binary SVM as the final output. In addition, the strategy allows us to design a parallel training process which trains different binary classifiers at the same time. In our SVM implementation, we modified the binary Tree Kernels in the SVM-Light Tool (SVM-Light-TK)[8]. It encodes the tree kernel inside the well known SVM-Light tool [Joachims 2002]. The parameters are tuned using the CoNLL-2005 development data set in the following experiments.

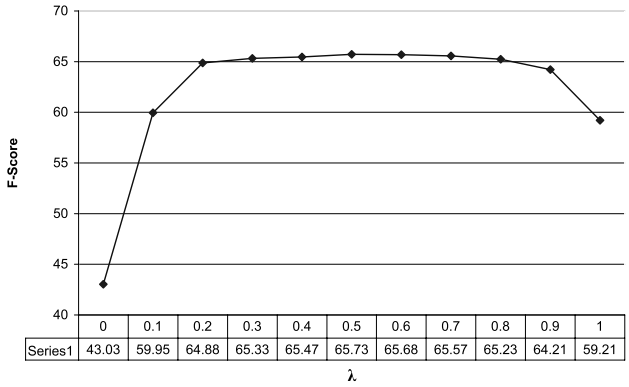
5.2 Experimental Results and Discussion

In order to speed up the tuning process for choosing an optimal setting of parameters, in the following experiments, we only use *WSJ* sections 02-05 of the CoNLL-2005 SRL shared task corpus as the training data and fine-tune the parameters on the CoNLL-2005 development set. Finally, we report the performance of using the entire training data set (including the CoNLL-2005 and the CPB data) and the fine-tuned parameters. In the same way as in Moschitti [2004], we also set the tree kernel decay factor $\mu = 0.4$ in the computation of convolution tree kernels.

The performance curve on the CoNLL-2005 development set with respect to λ (the weight of hybrid convolution tree kernel in Equation (1)) is shown in Figure 12. Here, we use the default SVM parameter setting in the SVM-Light.

Figure 12 shows that when $\lambda = 0.5$, the hybrid convolution tree kernel gets the best performance, $F_{\beta=1} = 65.73$. Both the Path kernel ($\lambda = 1$, $F_{\beta=1} = 59.21$) and the Constituent Structure kernel ($\lambda = 0$, $F_{\beta=1} = 43.03$) do not perform better than the hybrid one. It suggests that the two individual kernels are complementary to each other. This is the reason why we decompose the PAF kernel into a Path kernel and a Constituent Structure kernel. In addition,

[8]http://ai-nlp.info.uniroma2.it/moschitti/TK1.2-software/Tree-Kernel.htm



	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Series1	43.03	59.95	64.88	65.33	65.47	65.73	65.68	65.57	65.23	64.21	59.21

Fig. 12. The performance changing with λ in the hybrid convolution tree kernel with default SVM parameter setting.

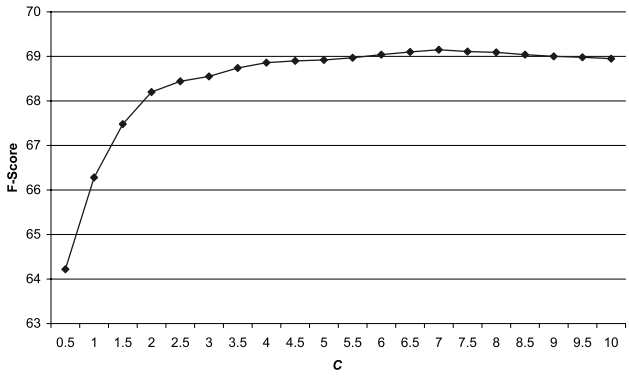


Fig. 13. The performance curve changing with SVM parameter C with default $e = 0.001$ in the hybrid convolution tree kernel.

the Path kernel performs much better than the Constituent Structure kernel. Through the tuning of parameter λ, we can decrease the influence of the Constituent Structure kernel and optimize the contribution of different kernels. We set the parameter $\lambda = 0.5$. The two individual kernels (K_{path} and K_{cs}) are normalized, respectively. It is different from the PAF kernel, where the Constituent Structure kernel usually dominates the final kernel value as shown in Figure 10. On the other hand, our hybrid convolution tree kernel emphasizes the contribution of the Path kernel more than the PAF kernel.

Figure 13 studies the effect of the parameter C in SVM on the hybrid convolution tree kernel. The parameter C controls the trade off between tolerating training errors and forcing rigid margins. It creates a soft margin on either side to allow for some misclassifications. Increasing the value of C increases the cost of misclassifying points and forces the creation of a more accurate model that may not generalize well.

We can see that the parameter C has a great influence on the performance of the hybrid convolution tree kernel. The performance improved sharply when C is increased initially. However, the rate gradually slows down until

Table III. The English PropBank Performance ($F_{\beta=1}$) Comparison Among the Hybrid Convolution Tree Kernel (Hybrid), the PAF Kernel, the MPAF Kernel, and Three Feature-Based Methods Using Linear Kernel and Polynomial Kernels $d = 2$ and $d = 3$. Parameter Cs and λ are Optimized on the Development Set for each Individual Method Respectively

	Hybrid ($C = 4.5$, $\lambda = 0.5$)	PAF ($C = 4$)	MPAF ($C = 4$)	Linear ($C = 2$)	Polynomial ($d = 2$, $C = 4$)	Polynomial ($d = 3$, $C = 3$)
Devel	68.90	67.03	67.80	69.36	72.63	72.28
Test (WSJ)	71.34	69.80	70.61	71.29	74.42	74.21
Test (Brown)	60.97	60.11	60.24	60.30	62.24	62.10

convergence. At the same time, the training time also takes longer because SVM has to find more exact hyperplanes. Therefore, considering the trade off between performance and training time, we select an optimal $C = 4.5$ for the hybrid convolution tree kernel. The optimal result ($F_{\beta=1} = 68.90$) is 3.17% higher than the default one ($F_{\beta=1} = 65.73$).

Table III compares the English PropBank performance among our hybrid convolution tree kernel, Moschitti [2004]'s PAF, MPAF kernel, and standard feature-based methods with linear and polynomial kernels ($d = 2, d = 3$) on the CoNLL-2005 development, and test data (*WSJ* section 23 and Brown corpus). Here, the *WSJ* sections 02-05 is used as the training data. It is worth pointing out that the parameter Cs and λ, listed in Table III, are optimized on the CoNLL-2005 development set for each individual method, respectively.

We can see that our hybrid convolution tree kernel outperforms the PAF kernel with statistically significantly[9] (χ^2 test with $p = 0.05$) on all development and test data. In addition, although the MPAF kernel also outperforms the PAF kernel, its performance is still significantly (χ^2 test with $p = 0.05$) worse than our hybrid convolution tree kernel. This empirically demonstrates that our hybrid kernel is more effective than the PAF and the MPAF kernels for SRL. In addition, comparison of columns (1) and (4) shows that using only the hybrid convolution tree kernel method, we can achieve a comparable performance with the feature-based method using the linear kernel (Linear). It means that if the syntactic structure can be modeled effectively, it is sufficiently competitive to other methods which use a large amount of diverse features.

However, our hybrid kernel still performs worse than the standard feature-based methods which use the polynomial kernel. This is simple because our kernel only use the syntactic structure information while the feature-based method uses a large number of hand-crafted diverse features, including word, POS, voice, etc., and especially combination of these features. The feature-based method with polynomial kernel ($d = 2$) achieves the best performance. It suggests that the binary combination among features implemented using the polynomial kernel ($d = 2$) is very useful. Therefore, we expect that the performance would be better by combining our hybrid convolution tree kernel with the polynomial kernel.

[9]We conduct a χ^2 test of significance to determine whether the difference in number of responses over all the confusion categories (correct, wrong, false positive, and false negative) are statistically significant at p.

Table IV. The Chinese PropBank Performance ($F_{\beta=1}$) Comparison Among the Hybrid Convolution Tree Kernel (Hybrid), the PAF Kernel, the MPAF Kernel, the Standard Feature-Based Method Using the Polynomial Kernels $d = 2$, the Composite Kernel, and the [Xue and Palmer's 2005] Work. The Parameters are Set as the Corresponding English Methods

	Hybrid ($C = 4.5$, $\lambda = 0.5$)	PAF ($C = 4$)	MPAF ($C = 4$)	Polynomial ($d = 2$, $C = 4$)	Xue and Palmer [2005]	Composite ($C = 4$, $\gamma = 0.2$)
Gold	85.85	84.43	84.77	91.13	91.3	**91.67**
Auto	60.12	58.83	59.21	64.79	61.3	**65.42**

Table IV compares the Chinese PropBank (CPB) performance with different methods. They are the hybrid convolution tree kernel (Hybrid), the PAF kernel, the MPAF kernel, the best performance feature-based method using the polynomial kernels $d = 2$, a composite kernel (which will be introduced later), and the Xue and Palmer's [2005] work. The Chinese flat features are imported from English.

The performance trends based on the handcrafted (gold) parse trees and the auto-parsed[10] trees are consistent, that is, our hybrid convolution tree kernel outperforms the PAF kernel and the MPAF kernel with statistical significance (χ^2 test with $p = 0.05$). For the same reason as above, our hybrid kernel performs worse than the standard feature-based method which uses the polynomial kernel ($d = 2$). The standard method achieves a comparable performance to Xue and Palmer's [2005] work, although the latter used a different classifier, maximum entropy [Berger et al. 1996], and a different feature set [Xue and Palmer 2005]. In addition, we can see that the polynomial kernel significantly outperforms Xue and Palmer's [2005] work based on the auto parsing, although they used a similar method, that is, feature-based method. We think that the main reason is that we use a better syntactic parser.

In order to make full use of the syntactic information and the standard flat features, we present a composite kernel to combine the hybrid convolution tree kernel (K_{hybrid}) with a feature-based method with polynomial kernel (K_{poly}):

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

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

The performance with respect to γ on the development set of the CoNLL 2005 share task is shown in Figure 14. Here, note that we use the polynomial kernel ($d = 2$) with a default C in SVM.

We can see that when $\gamma = 0.2$, the system achieves the best performance with $F_{\beta=1} = \mathbf{70.74}$. It is a statistically significant improvement (χ^2 test with $p = 0.1$) over using only the feature-based method with the polynomial kernel ($\gamma = 0$, $F_{\beta=1} = 70.43$) and much higher than using only the hybrid convolution tree kernel ($\gamma = 1$, $F_{\beta=1} = 65.73$)[11]. The main reason is that the convolu-

[10]Dan Bikel's multilingual statistical parsing engine is used. Similar to Xue and Palmer [2005], the parser is trained on all the data in the Penn Chinese Treebank except for the test data that has been set aside. A word segmentation system (http://ir.hit.edu.cn/demo/ltp/) is used on the test data before parsing.

[11]Note that all the results are gotten with default C. So the performances are different from that in Table III where Cs are optimized.

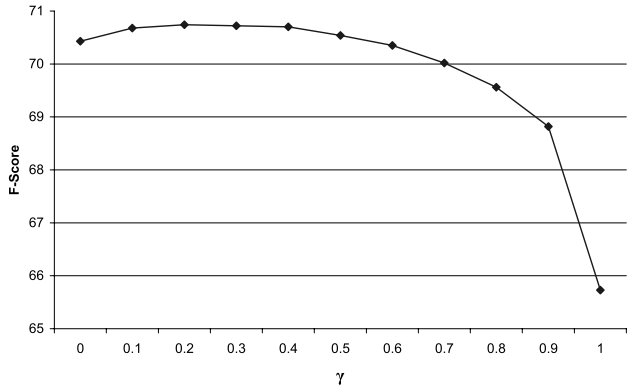


Fig. 14. The performance with respect to γ in the composite kernel with default SVM parameter setting.

Table V. Performance ($F_{\beta=1}$) Comparison Among the Composite Kernel (the Hybrid Convolution Tree Kernel + the Polynomial Kernel (d = 2)), the Feature-Based Method with Polynomial Kernel (d = 2), and the Best Reported System in the CoNLL-2005 SRL Shared Task when Using Only One Syntactic Parser

	Composite (C = 4)	Polynomial (d = 2, C = 4)	Surdeanu and Turmo [2005] (the best one with single parser)
Devel	75.66	75.37	75.17
Test (WSJ)	77.41	77.00	76.46
Test (Brown)	66.21	65.63	65.42

tion tree kernel based methods can represent more general syntactic features than standard feature-based methods. On the other hand, the feature-based method captures more features than what the convolution tree kernel-based method can represent, such as Voice, Named Entity. Thus the two methods are complementary to each other.

To find the optimal parameters for the composite kernel, we tune C again and find the optimal C = 4.

Finally, we train the composite kernel and the polynomial kernel (d = 2) using the above parameter setting (i.e., λ = 0.5, γ = 0.2, and default e = 0.001, Cs are optimized for each individual methods) on the entire CoNLL-2005 SRL shared task training set (WSJ sections 02-21). Table V compares the performance among the composite kernel, the polynomial kernel, and a CoNLL-2005 SRL shared task system Surdeanu and Turmo [2005], which ranks fifth among all participating systems in the shared task, but the best one when a single syntactic parser Charniak [2000] is used (using the same parse strategy as ours). However, they used a different classifier, AdaBoost [Schapire and Singer 1999], and a different feature set [Surdeanu and Turmo 2005].

Comparison of columns (2) and (3) shows that using the SVM classifier with the polynomial kernel (d = 2) outperforms Surdeanu and Turmo's [2005] AdaBoost. However, the improvement is not significant. On the other hand, the composite kernel improves about 0.3% $\sim$ 0.6% over the polynomial kernel

Table VI. Comparison of Computational Time

Classification Methods	Training Time		Test Time
	4 Sections	20 Sections	
Feature-based	∼3 hours	∼2 days	5 min
Hybrid Convolution Tree Kernel	∼5 hours	∼6 days	10 min

(columns 1 vs. 2) and outperforms the best reported system in the CoNLL-2005 SRL shared task using the same parse results (columns 1 vs. 3).

Similar to the English experiments, we train the composite kernel for the CPB using the same parameter setting as English. With the final $F_{\beta=1}$ is **91.67** as shown in Table IV, the composite kernel outperforms the feature-based methods with the polynomial kernel ($F_{\beta=1}$ = 91.13) and Xue and Palmer's [2005] work ($F_{\beta=1}$ = 91.3) respectively.

The above experiments on English and Chinese further verify the effectiveness of the hybrid convolution tree kernel method for SRL.

Finally, Table VI compares the computational time of the standard feature-based and our hybrid convolution tree kernel with the same SVM kernel machine on CoNLL 2005 dataset (2.0GHz×2 Xeon CPU and 4G Memory). It shows that:

(1) The tree kernel is slower than the standard feature-based methods.

(2) It is very time-consuming to train an SVM classifier in a large dataset.

6. CONCLUSIONS AND FUTURE WORK

In this article, we have proposed a hybrid convolution tree kernel to model syntactic structure information for semantic role labeling (SRL). Different from the previous convolution tree kernel based methods, we distinguish the Path and the Constituent Structure feature spaces. Evaluations on the data sets of the CoNLL-2005 SRL shared task and Chinese PropBank (CPB) show that our novel hybrid convolution tree kernel significantly outperforms the previous predicate argument feature (PAF) kernel and its improved version (MPAF). Therefore, we suggest the approach of using multiple tree kernels to model different linguistic objects in natural language processing applications. The final composite kernel between the hybrid convolution tree kernel method and the feature-based method with the polynomial kernel (d = 2) outperforms the best reported systems on CoNLL-2005 corpus using a single parser and on the CPB corpus using correct syntactic parse results respectively.

The immediate extension for our work is to integrate more linguistic knowledge in convolution tree kernels. For example, we can do approximate substructure matching based on linguistic knowledge. We can also do feature selection under convolution tree kernel framework with linguistic knowledge. Moreover, we can also explore the hybrid convolution tree kernel method in other tasks, such as relation extraction in the future.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their critical and insightful comments.

REFERENCES

BAKER, C. F., FILLMORE, C. J., AND LOWE, J. B. 1998. The Berkeley FrameNet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (COLING-ACL'98)*, 86–90.

BERGER, A. L., DELLA PIETRA, S. A., AND DELLA PIETRA, V. J. 1996. A maximum entropy approach to natural language processing. *Comput. Linguist. 22*, 1, 39–71.

CARRERAS, X. AND MÀRQUEZ, L. 2004. Introduction to the CoNLL-2004 shared task: Semantic role labeling. In *Proceedings of the 8th Conference on Natural Language Learning (CoNLL'04)*. 89–97.

CARRERAS, X. AND MÀRQUEZ, L. 2005. Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *Proceedings of the 9th Conference on Natural Language Learning (CoNLL'05)*. 152–164.

CHARNIAK, E. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st Conference of the North American Chapter of the Association for Computational Linguistics (NAACL'00)*.

CHE, W., ZHANG, M., LIU, T., AND LI, S. 2006. A hybrid convolution tree kernel for semantic role labeling. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics and 21st International Conference on Computational Linguistics (COLING-ACL'06)*. Sydney, Australia.

CHIEU, H. L. AND NG, H. T. 2003. Named entity recognition with a maximum entropy approach. In *Proceedings of the 7th Conference on Natural Language Learning (CoNLL'03)*. 160–163.

COLLINS, M. 1999. Head-driven statistical models for natural language parsing. Ph.D. thesis, Pennsylvania University.

COLLINS, M. AND DUFFY, N. 2001. Convolution kernels for natural language. In *Proceedings of the 15th Annual Conference on Neutral Information Processing Systems (NIPS'01)*.

CRISTIANINI, N. AND SHAWE-TAYLOR, J. 2000. *An Introduction to Support Vector Machines*. Cambridge University Press, New York.

CULOTTA, A. AND SORENSEN, J. 2004. Dependency tree kernels for relation extraction. In *Proceedings of the 42th Annual Meeting of the Association for Computer Linguistics (ACL'04)*. 423–429.

GILDEA, D. AND JURAFSKY, D. 2002. Automatic labeling of semantic roles. *Comput. Linguist. 28*, 3, 245–288.

GILDEA, D. AND PALMER, M. 2002. The necessity of parsing for predicate argument recognition. In *Proceedings of the 40th Anniversary Meeting of the Association for Computer Linguistics (ACL'02)*. 239–246.

GIMENEZ, J. AND MÀRQUEZ, L. 2003. Fast and accurate part-of-speech tagging: The svm approach revisited. In *Proceedings of the International Conference on Recent Advances in Natural Language (RANLP'03)*.

HAUSSLER, D. 1999. Convolution kernels on discrete structures. Tech. Rep. UCSC-CRL-99-10. July.

JIANG, Z. P., LI, J., AND NG, H. T. 2005. Semantic argument classification exploiting argument interdependence. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI'05)*.

JOACHIMS, T. 2002. *Learning to Classify Text Using Support Vector Machines: Methods, Theory and Algorithms*. Kluwer Academic Publishers, Norwell, MA.

JOACHIMS, T., CRISTIANINI, N., AND SHAWE-TAYLOR, J. 2001. Composite kernels for hypertext categorization. In *Proceedings of the 18th International Conference on Machine Learning (ICML'01)*. 250–257.

LIU, T., CHE, W., LI, S., HU, Y., AND LIU, H. 2005. Semantic role labeling system using maximum entropy classifier. In *Proceedings of the 9th Conference on Natural Language Learning (CoNLL'05)*. 189–192.

LODHI, H., SAUNDERS, C., SHAWE-TAYLOR, J., CRISTIANINI, N., AND WATKINS, C. 2002. Text classification using string kernels. *J. Mach. Learn. Res. 2*, 419–444.

MARCUS, M. P., MARCINKIEWICZ, M. A., AND SANTORINI, B. 1993. Building a large annotated corpus of English: the penn treebank. *Comput. Linguist. 19*, 2, 313–330.

MOSCHITTI, A. 2004. A study on convolution kernels for shallow statistic parsing. In *Proceedings of the 42nd Annual Meeting of the Association for Computer Linguistics (ACL'04)*. 335–342.

MOSCHITTI, A., PIGHIN, D., AND BASILI, R. 2006. Tree kernel engineering in semantic role labeling systems. In *Proceedings of the Workshop on Learning Structured Information for Natural Language Applications, 11th International Conference on European Association for Computational Linguistics (EACL'06)*. Trento, Italy, 49–56.

MOSCHITTI, A., PIGHIN, D., AND BASILI, R. To appear. Tree kernels for semantic role labeling. *Comput. Linguist.* (forthcoming).

MOSCHITTI, A., QUARTERONI, S., BASILI, R., AND MANANDHAR, S. 2006. Exploiting syntactic and shallow semantic kernels for question answer classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL'07)*. Prague, Czech Republic, 776–783.

NARAYANAN, S. AND HARBABAGIU, S. 2004. Question answering based on semantic structures. In *Proceedings of the 20th International Conference on Computer Linguistics (COLING'04)*.

NIELSEN, R. D. AND PRADHAN, S. 2004. Mixing weak learners in semantic parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'04)*.

PALMER, M., GILDEA, D., AND KINGSBURY, P. 2005. The proposition bank: An annotated corpus of semantic roles. *Comput. Linguist. 31*, 1, 71–106.

PONZETTO, S. P. AND STRUBE, M. 2006. Exploiting semantic role labeling, wordnet and wikipedia for coreference resolution. In *Proceedings of the Loint Human Language Technology Conference/Annula Meeting of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL'06)*. New York.

PRADHAN, S., HACIOGLU, K., KRUGLER, V., WARD, W., MARTIN, J. H., AND JURAFSKY, D. 2005. Support vector learning for semantic argument classification. *Mach. Learn.*

PRADHAN, S., WARD, W., HACIOGLU, K., MARTIN, J., AND JURAFSKY, D. 2005. Semantic role labeling using different syntactic views. In *Proceedings of the 43rd Annual Meeting of the Association for Computer Linguistics (ACL'05)*. 581–588.

PUNYAKANOK, V., ROTH, D., AND TAU YIH, W. 2005. The necessity of syntactic parsing for semantic role labeling. In *Proceedings of 19th International Joint Conference on Artificial Intelligence (IJCAI'05)*. 1117–1123.

PUNYAKANOK, V., ROTH, D., YIH, W.-T., AND ZIMAK, D. 2004. Semantic role labeling via integer linear programming inference. In *Proceedings of the 20th International Conference on Computer Linguistics (COLING'04)*. 1346–1352.

RIFKIN, R. AND KLAUTAU, A. 2004. In defense of one-vs-all classification. *J. Mach. Learn. Res. 5*, 101–141.

SCHAPIRE, R. E. AND SINGER, Y. 1999. Improved boosting algorithms using confidence-rated predictions. *Mach. Learn. 37*, 3, 297–336.

SHAWE-TAYLOR, J. AND CRISTIANINI, N. 2004. *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York.

SHEN, D. AND LAPATA, M. 2007. Using semantic roles to improve question answering. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL'07)*. 12–21.

SUN, H. AND JURAFSKY, D. 2004. Shallow semantic parsing of chinese. In *Proceedings of the Joint Human Language Technology Conference/Annual Meeting of the North American Chapter of the Association for Computational Linguistics (HLT/NAACL'04)*.

SURDEANU, M., HARABAGIU, S., WILLIAMS, J., AND AARSETH, P. 2003. Using predicate-argument structures for information extraction. In *Proceedings of the 41st Annual Meeting of the Association for Computer Linguistics (ACL'03)*.

SURDEANU, M. AND TURMO, J. 2005. Semantic role labeling using complete syntactic analysis. In *Proceedings of the 9th Conference on Natural Language Learning (CoNLL'05)*. Ann Arbor, Michigan.

VAPNIK, V. N. 1998. *Statistical learning theory*. Wiley.

WATKINS, C. 1999. Dynamic alignment kernels. Tech. Rep. CSD-TR-98-11. January.

XUE, N. AND KULICK, S. 2003. Automatic predicate argument structure analysis of the penn Chinese treebank. In *Proceedings of the 10th Machine Translation Summit (MT Summit X)*.

XUE, N. AND PALMER, M. 2004. Calibrating features for semantic role labeling. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'04)*.

XUE, N. AND PALMER, M. 2005. Automatic semantic role labeling for Chinese verbs. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI'05)*.

XUE, N., XIA, F., DONG CHIOU, F., AND PALMER, M. 2005. The Penn Chinese Treebank: Phrase structure annotation of a large corpus. *Nat. Lang. Eng. 11*, 2, 207–238.

ZELENKO, D., AONE, C., AND RICHARDELLA, A. 2003. Kernel methods for relation extraction. *J. Mach. Learn. Res. 3*, 1083–1106.

ZHANG, M., ZHANG, J., AND SU, J. 2006a. Exploring syntactic features for relation extraction using a convolution tree kernel. In *Proceedings of the Joint Human Language Technology Conference/Annual Meeting of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL'06)*. New York City.

ZHANG, M., ZHANG, J., SU, J., AND ZHOU, G. 2006b. A composite kernel to extract relations between entities with both flat and structured features. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics and 21st International Conference on Computational Linguistics (COLING/ACL'06)*. Sydney, Australia.

Received April 2007; revised April 2008, July 2008; accepted August 2008