A Grammar-driven Convolution Tree Kernel for Semantic Role Classification

Min ZHANG[1] **Wanxiang CHE**[2] **Ai Ti AW**[1] **Chew Lim TAN**[3]
Guodong ZHOU[1,4] **Ting LIU**[2] **Sheng LI**[2]

[1]Institute for Infocomm Research
{mzhang, aaiti}@i2r.a-star.edu.sg

[2]Harbin Institute of Technology
{car, tliu}@ir.hit.edu.cn
lisheng@hit.edu.cn

[3]National University of Singapore
tancl@comp.nus.edu.sg

[4] Soochow Univ., China 215006
gdzhou@suda.edu.cn

Abstract

Convolution tree kernel has shown promising results in semantic role classification. However, it only carries out hard matching, which may lead to over-fitting and less accurate similarity measure. To remove the constraint, this paper proposes a grammar-driven convolution tree kernel for semantic role classification by introducing more linguistic knowledge into the standard tree kernel. The proposed grammar-driven tree kernel displays two advantages over the previous one: 1) grammar-driven approximate substructure matching and 2) grammar-driven approximate tree node matching. The two improvements enable the grammar-driven tree kernel explore more linguistically motivated structure features than the previous one. Experiments on the CoNLL-2005 SRL shared task show that the grammar-driven tree kernel significantly outperforms the previous non-grammar-driven one in SRL. Moreover, we present a composite kernel to integrate feature-based and tree kernel-based methods. Experimental results show that the composite kernel outperforms the previously best-reported methods.

1 Introduction

Given a sentence, the task of Semantic Role Labeling (SRL) consists of analyzing the logical forms expressed by some target verbs or nouns and some constituents of the sentence. In particular, for each predicate (target verb or noun) all the constituents in the sentence which fill semantic arguments (roles) of the predicate have to be recognized. Typical semantic roles include *Agent, Patient, Instrument,* etc. and also adjuncts such as *Locative, Temporal, Manner,* and *Cause,* etc. Generally, semantic role identification and classification are regarded as two key steps in semantic role labeling. Semantic role identification involves classifying each syntactic element in a sentence into either a semantic argument or a non-argument while semantic role classification involves classifying each semantic argument identified into a specific semantic role. This paper focuses on semantic role classification task with the assumption that the semantic arguments have been identified correctly.

Both feature-based and kernel-based learning methods have been studied for semantic role classification (Carreras and Màrquez, 2004; Carreras and Màrquez, 2005). In feature-based methods, a flat feature vector is used to represent a predicate-argument structure while, in kernel-based methods, a kernel function is used to measure directly the similarity between two predicate-argument structures. As we know, kernel methods are more effective in capturing structured features. Moschitti (2004) and Che et al. (2006) used a convolution tree kernel (Collins and Duffy, 2001) for semantic role classification. The convolution tree kernel takes sub-tree as its feature and counts the number of common sub-trees as the similarity between two predicate-arguments. This kernel has shown very

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 200–207,
Prague, Czech Republic, June 2007. ©2007 Association for Computational Linguistics

promising results in SRL. However, as a general learning algorithm, the tree kernel only carries out hard matching between any two sub-trees without considering any linguistic knowledge in kernel design. This makes the kernel fail to handle similar phrase structures (e.g., “buy a car” *vs*. “buy a *red* car”) and near-synonymic grammar tags (e.g., the POS variations between “*high*/JJ degree/NN” [1] and “*higher*/JJR degree/NN”) [2]. To some degree, it may lead to over-fitting and compromise performance.

This paper reports our preliminary study in addressing the above issue by introducing more linguistic knowledge into the convolution tree kernel. To our knowledge, this is the first attempt in this research direction. In detail, we propose a grammar-driven convolution tree kernel for semantic role classification that can carry out more linguistically motivated substructure matching. Experimental results show that the proposed method significantly outperforms the standard convolution tree kernel on the data set of the CoNLL-2005 SRL shared task.

The remainder of the paper is organized as follows: Section 2 reviews the previous work and Section 3 discusses our grammar-driven convolution tree kernel. Section 4 shows the experimental results. We conclude our work in Section 5.

2 Previous Work

Feature-based Methods for SRL: most features used in prior SRL research are generally extended from Gildea and Jurafsky (2002), who used a linear interpolation method and extracted basic flat features from a parse tree to identify and classify the constituents in the FrameNet (Baker et al., 1998). Here, the basic features include Phrase Type, Parse Tree Path, and Position. Most of the following work 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 work paid much attention to the robust SRL (Pradhan et al., 2005b) and post inference (Punyakanok et al., 2004). These feature-based methods are considered as the state of the art methods for SRL. However, as we know, the standard flat features are less effective in modeling the syntactic structured information. For example, in SRL, the Parse Tree Path feature is sensitive to small changes of the syntactic structures. Thus, a predicate argument pair will have two different Path features even if their paths differ only for one node. This may result in data sparseness and model generalization problems.

Kernel-based Methods for SRL: as an alternative, kernel methods are more effective in modeling structured objects. This is because a kernel can measure the similarity between two structured objects using the original representation of the objects instead of explicitly enumerating their features. Many kernels have been proposed and applied to the NLP study. In particular, Haussler (1999) proposed the well-known convolution kernels for a discrete structure. In the context of it, more and more kernels for restricted syntaxes or specific domains (Collins and Duffy, 2001; Lodhi et al., 2002; Zelenko et al., 2003; Zhang et al., 2006) are proposed and explored in the NLP domain.

Of special interest here, Moschitti (2004) proposed Predicate Argument Feature (PAF) kernel for SRL under the framework of convolution tree kernel. He selected portions of syntactic parse trees as predicate-argument feature spaces, which include salient substructures of predicate-arguments, to define convolution kernels for the task of semantic role classification. Under the same framework, Che et al. (2006) proposed a hybrid convolution tree kernel, which consists of two individual convolution kernels: a Path kernel and a Constituent Structure kernel. Che et al. (2006) showed that their method outperformed PAF on the CoNLL-2005 SRL dataset.

The above two kernels are special instances of convolution tree kernel for SRL. As discussed in Section 1, convolution tree kernel only carries out hard matching, so it fails to handle similar phrase structures and near-synonymic grammar tags. This paper presents a grammar-driven convolution tree kernel to solve the two problems

3 Grammar-driven Convolution Tree Kernel

3.1 Convolution Tree Kernel

In convolution tree kernel (Collins and Duffy, 2001), a parse tree T is represented by a vector of integer counts of each sub-tree type (regardless of its ancestors): $\phi(T) = (\ \dots,\ \#subtree_i(T),\ \dots)$, where

[1] Please refer to http://www.cis.upenn.edu/~treebank/ for the detailed definitions of the grammar tags used in the paper.

[2] Some rewrite rules in English grammar are generalizations of others: for example, “NP→ DET JJ NN” is a specialized version of “NP→ DET NN”. The same applies to POS. The standard convolution tree kernel is unable to capture the two cases.

# *subtree$_i$(T)* is the occurrence number of the ith sub-tree type (*subtree$_i$*) in T. Since the number of different sub-trees is exponential with the parse tree size, it is computationally infeasible to directly use the feature vector $\phi(T)$. To solve this computational issue, Collins and Duffy (2001) proposed the following parse tree kernel to calculate the dot product between the above high dimensional vectors implicitly.

$$K(T_1,T_2) = <\phi(T_1),\phi(T_2)>$$
$$= \sum_i \left(\left(\sum_{n_1\in N_1} I_{subtree_i}(n_1)\right) \cdot \left(\sum_{n_2\in N_2} I_{subtree_i}(n_2)\right)\right)$$
$$= \sum_{n_1\in N_1}\sum_{n_2\in N_2}\Delta(n_1,n_2)$$

where N_1 and N_2 are the sets of nodes in trees T_1 and T_2, respectively, and $I_{subtree_i}(n)$ is a function that is 1 iff the *subtree$_i$* occurs with root at node n and zero otherwise, and $\Delta(n_1,n_2)$ is the number of the common *subtrees* rooted at n_1 and n_2, i.e.,

$$\Delta(n_1,n_2) = \sum_i I_{subtree_i}(n_1)\cdot I_{subtree_i}(n_2)$$

$\Delta(n_1,n_2)$ can be further computed efficiently by the following recursive rules:

Rule 1: if the productions (CFG rules) at n_1 and n_2 are different, $\Delta(n_1,n_2)=0$;

Rule 2: else if both n_1 and n_2 are pre-terminals (POS tags), $\Delta(n_1,n_2)=1\times\lambda$;

Rule 3: else,

$$\Delta(n_1,n_2) = \lambda\prod_{j=1}^{nc(n_1)}(1+\Delta(ch(n_1,j),ch(n_2,j))),$$

where $nc(n_1)$ is the child number of n_1, $ch(n,j)$ is the jth child of node n and λ (0< λ <1) is the decay factor in order to make the kernel value less variable with respect to the *subtree* sizes. In addition, the recursive **Rule 3** holds because given two nodes with the same children, one can construct common sub-trees using these children and common sub-trees of further offspring. The time complexity for computing this kernel is $O(|N_1|\cdot|N_2|)$.

3.2 Grammar-driven Convolution Tree Kernel

This Subsection introduces the two improvements and defines our grammar-driven tree kernel.

Improvement 1: Grammar-driven approximate matching between substructures. The conventional tree kernel requires exact matching between two contiguous phrase structures. This constraint may be too strict. For example, the two phrase structures "NP→DT JJ NN" (NP→*a red car*) and "NP→DT NN" (NP->*a car*) are not identical, thus they contribute nothing to the conventional kernel although they should share the same semantic role given a predicate. In this paper, we propose a grammar-driven approximate matching mechanism to capture the similarity between such kinds of quasi-structures for SRL.

First, we construct reduced rule set by defining optional nodes, for example, "NP->DT [JJ] NP" or "VP-> VB [ADVP] PP", where [*] denotes optional nodes. For convenience, we call "NP-> DT JJ NP" the original rule and "NP->DT [JJ] NP" the reduced rule. Here, we define two grammar-driven criteria to select optional nodes:

1) The reduced rules must be grammatical. It means that the reduced rule should be a valid rule in the original rule set. For example, "NP->DT [JJ] NP" is valid only when "NP->DT NP" is a valid rule in the original rule set while "NP->DT [JJ NP]" may not be valid since "NP->DT" is not a valid rule in the original rule set.

2) A valid reduced rule must keep the head child of its corresponding original rule and has at least two children. This can make the reduced rules retain the underlying semantic meaning of their corresponding original rules.

Given the reduced rule set, we can then formulate the approximate substructure matching mechanism as follows:

$$M(r_1,r_2) = \sum_{i,j}(I_T(T_{r1}^i,T_{r2}^j)\times\lambda_1^{a_i+b_j}) \qquad (1)$$

where r_1 is a production rule, representing a sub-tree of depth one[3], and T_{r1}^i is the ith variation of the sub-tree r_1 by removing one ore more optional nodes[4], and likewise for r_2 and T_{r2}^j. $I_T(\bullet,\bullet)$ is a function that is 1 iff the two sub-trees are identical and zero otherwise. λ_1 (0≤ λ_1 ≤1) is a small penalty to penal-

[3] Eq.(1) is defined over sub-structure of depth one. The approximate matching between structures of depth more than one can be achieved easily through the matching of sub-structures of depth one in the recursively-defined convolution kernel. We will discuss this issue when defining our kernel.

[4] To make sure that the new kernel is a proper kernel, we have to consider all the possible variations of the original sub-trees. Training program converges only when using a proper kernel.

ize optional nodes and the two parameters a_i and b_j stand for the numbers of occurrence of removed optional nodes in subtrees T_{r1}^i and T_{r2}^j, respectively. $M(r_1, r_2)$ returns the similarity (ie., the kernel value) between the two sub-trees r_1 and r_2 by summing up the similarities between all possible variations of the sub-trees r_1 and r_2.

Under the new approximate matching mechanism, two structures are matchable (but with a small penalty λ_1) if the two structures are identical after removing one or more optional nodes. In this case, the above example phrase structures "NP->*a red car*" and "NP->*a car*" are matchable with a penalty λ_1 in our new kernel. It means that one co-occurrence of the two structures contributes λ_1 to our proposed kernel while it contributes zero to the traditional one. Therefore, by this improvement, our method would be able to explore more linguistically appropriate features than the previous one (which is formulated as $I_T(r_1, r_2)$).

Improvement 2: Grammar-driven tree nodes approximate matching. The conventional tree kernel needs an exact matching between two (terminal/non-terminal) nodes. But, some similar POSs may represent similar roles, such as NN (*dog*) and NNS (*dogs*). In order to capture this phenomenon, we allow approximate matching between node features. The following illustrates some equivalent node feature sets:

- JJ, JJR, JJS
- VB, VBD, VBG, VBN, VBP, VBZ
- ……

where POSs in the same line can match each other with a small penalty $0 \le \lambda_2 \le 1$. We call this case node feature *mutation*. This improvement further generalizes the conventional tree kernel to get better coverage. The approximate node matching can be formulated as:

$$M(f_1, f_2) = \sum_{i,j} (I_f(f_1^i, f_2^j) \times \lambda_2^{a_i+b_j}) \qquad (2)$$

where f_1 is a node feature, f_1^i is the i^{th} mutation of f_1 and a_i is 0 iff f_1^i and f_1 are identical and 1 otherwise, and likewise for f_2. $I_f(\bullet, \bullet)$ is a function that is 1 iff the two features are identical and zero otherwise. Eq. (2) sums over all combinations of feature mutations as the node feature similarity. The same as Eq. (1), the reason for taking all the possibilities into account in Eq. (2) is to make sure that the new kernel is a proper kernel.

The above two improvements are grammar-driven, i.e., the two improvements retain the underlying linguistic grammar constraints and keep semantic meanings of original rules.

The Grammar-driven Kernel Definition: Given the two improvements discussed above, we can define the new kernel by beginning with the feature vector representation of a parse tree T as follows:

$$\phi'(T) = (\#subtree_1(T), \ldots, \#subtree_n(T))$$

where $\#subtree_i(T)$ is the occurrence number of the i^{th} sub-tree type ($subtree_i$) in T. Please note that, different from the previous tree kernel, here we loosen the condition for the occurrence of a subtree by allowing both original and reduced rules (**Improvement 1**) and node feature mutations (**Improvement 2**). In other words, we modify the criteria by which a subtree is said to occur. For example, one occurrence of the rule "NP->DT JJ NP" shall contribute 1 times to the feature "NP->DT JJ NP" and λ_1 times to the feature "NP->DT NP" in the new kernel while it only contributes 1 times to the feature "NP->DT JJ NP" in the previous one. Now we can define the new grammar-driven kernel $K_G(T_1, T_2)$ as follows:

$$\begin{aligned} K_G(T_1, T_2) &= <\phi'(T_1), \phi'(T_2)> \\ &= \sum_i \left(\left(\sum_{n_1 \in N_1} I'_{subtree_i}(n_1)\right) \cdot \left(\sum_{n_2 \in N_2} I'_{subtree_i}(n_2)\right) \right) \qquad (3) \\ &= \sum_{n_1 \in N_1} \sum_{n_2 \in N_2} \Delta'(n_1, n_2) \end{aligned}$$

where N_1 and N_2 are the sets of nodes in trees T_1 and T_2, respectively. $I'_{subtree_i}(n)$ is a function that is $\lambda_1^a \bullet \lambda_2^b$ iff the $subtree_i$ occurs with root at node n and zero otherwise, where a and b are the numbers of removed optional nodes and mutated node features, respectively. $\Delta'(n_1, n_2)$ is the number of the common *subtrees* rooted at n_1 and n_2, i.e. ,

$$\Delta'(n_1, n_2) = \sum_i I'_{subtree_i}(n_1) \cdot I'_{subtree_i}(n_2) \qquad (4)$$

Please note that the value of $\Delta'(n_1, n_2)$ is no longer an integer as that in the conventional one since optional nodes and node feature mutations are considered in the new kernel. $\Delta'(n_1, n_2)$ can be further computed by the following recursive rules:

===

Rule A: if n_1 and n_2 are pre-terminals, then:

$$\Delta'(n_1,n_2)=\lambda\times M(f_1,f_2) \tag{5}$$

where f_1 and f_2 are features of nodes n_1 and n_2 respectively, and $M(f_1,f_2)$ is defined at Eq. (2).

Rule B: else if both n_1 and n_2 are the same non-terminals, then generate all variations of the subtrees of *depth one* rooted by n_1 and n_2 (denoted by T_{n1} and T_{n2} respectively) by removing different optional nodes, then:

$$\Delta'(n_1,n_2)=\lambda\times\sum\nolimits_{i,j}(I_T(T_{n1}^i,T_{n2}^j)\times\lambda_1^{a_i+b_j} \times\prod\nolimits_{k=1}^{nc(n_1,i)}(1+\Delta'(ch(n_1,i,k),ch(n_2,j,k))) \tag{6}$$

where

- T_{n1}^i and T_{n2}^j stand for the i^{th} and j^{th} variations in sub-tree set T_{n1} and T_{n2}, respectively.
- $I_T(\bullet,\bullet)$ is a function that is 1 iff the two sub-trees are identical and zero otherwise.
- a_i and b_j stand for the number of removed optional nodes in subtrees T_{n1}^i and T_{n2}^j, respectively.
- $nc(n_1,i)$ returns the child number of n_1 in its i^{th} subtree variation T_{n1}^i.
- $ch(n_1,i,k)$ is the k^{th} child of node n_1 in its i^{th} variation subtree T_{n1}^i, and likewise for $ch(n_2,j,k)$.
- Finally, the same as the previous tree kernel, λ ($0<\lambda<1$) is the decay factor (see the discussion in Subsection 3.1).

Rule C: else $\Delta'(n_1,n_2)=0$

===

Rule A accounts for **Improvement** 2 while **Rule B** accounts for **Improvement** 1. In **Rule B**, Eq. (6) is able to carry out multi-layer sub-tree approximate matching due to the introduction of the recursive part while Eq. (1) is only effective for subtrees of depth one. Moreover, we note that Eq. (4) is a convolution kernel according to the definition and the proof given in Haussler (1999), and Eqs (5) and (6) reformulate Eq. (4) so that it can be computed efficiently, in this way, our kernel defined by Eq (3) is also a valid convolution kernel. Finally, let us study the computational issue of the new convolution tree kernel. Clearly, computing Eq. (6) requires exponential time in its worst case. However, in practice, it may only need $O(|N_1|\cdot|N_2|)$. This is because there are only 9.9% rules (647 out of the total 6,534 rules in the parse trees) have optional nodes and most of them have only one optional node. In fact, the actual running time is even much less and is close to linear in the size of the trees since $\Delta'(n_1,n_2)=0$ holds for many node pairs (Collins and Duffy, 2001). In theory, we can also design an efficient algorithm to compute Eq. (6) using a dynamic programming algorithm (Moschitti, 2006). We just leave it for our future work.

3.3 Comparison with previous work

In above discussion, we show that the conventional convolution tree kernel is a special case of the grammar-driven tree kernel. From kernel function viewpoint, our kernel can carry out not only exact matching (as previous one described by **Rules 2** and **3** in Subsection 3.1) but also approximate matching (Eqs. (5) and (6) in Subsection 3.2). From feature exploration viewpoint, although they explore the same sub-structure feature space (defined recursively by the phrase parse rules), their feature values are different since our kernel captures the structure features in a more linguistically appropriate way by considering more linguistic knowledge in our kernel design.

Moschitti (2006) proposes a partial tree (PT) kernel which can carry out partial matching between sub-trees. The PT kernel generates a much larger feature space than both the conventional and the grammar-driven kernels. In this point, one can say that the grammar-driven tree kernel is a specialization of the PT kernel. However, the important difference between them is that the PT kernel is not grammar-driven, thus many non-linguistically motivated structures are matched in the PT kernel. This may potentially compromise the performance since some of the over-generated features may possibly be noisy due to the lack of linguistic interpretation and constraint.

Kashima and Koyanagi (2003) proposed a convolution kernel over labeled order trees by generalizing the standard convolution tree kernel. The labeled order tree kernel is much more flexible than the PT kernel and can explore much larger sub-tree features than the PT kernel. However, the same as the PT kernel, the labeled order tree kernel is not grammar-driven. Thus, it may face the same issues (such as over-generated features) as the PT kernel when used in NLP applications.

Shen el al. (2003) proposed a lexicalized tree kernel to utilize LTAG-based features in parse reranking. Their methods need to obtain a LTAG derivation tree for each parse tree before kernel calculation. In contrast, we use the notion of optional arguments to define our grammar-driven tree kernel and use the empirical set of CFG rules to determine which arguments are optional.

4 Experiments

4.1 Experimental Setting

Data: We use the CoNLL-2005 SRL shared task data (Carreras and Màrquez, 2005) as our experimental corpus. The data consists of sections of the Wall Street Journal part of the Penn TreeBank (Marcus et al., 1993), with information on predicate-argument structures extracted from the PropBank corpus (Palmer et al., 2005). As defined by the shared task, we use sections 02-21 for training, section 24 for development and section 23 for test. There are 35 roles in the data including 7 Core (A0–A5, AA), 14 Adjunct (AM-) and 14 Reference (R-) arguments. Table 1 lists counts of sentences and arguments in the three data sets.

	Training	Development	Test
Sentences	39,832	1,346	2,416
Arguments	239,858	8,346	14,077

Table 1: Counts on the data set

We assume that the semantic role identification has been done correctly. In this way, we can focus on the classification task and evaluate it more accurately. We evaluate the performance with Accuracy. SVM (Vapnik, 1998) is selected as our classifier and the *one vs. others* strategy is adopted and the one with the largest margin is selected as the final answer. In our implementation, we use the binary SVMLight (Joachims, 1998) and modify the Tree Kernel Tools (Moschitti, 2004) to a grammar-driven one.

Kernel Setup: We use the Constituent, Predicate, and Predicate-Constituent related features, which are reported to get the best-reported performance (Pradhan et al., 2005a), as the baseline features. We use Che et al. (2006)'s hybrid convolution tree kernel (the best-reported method for kernel-based SRL) as our baseline kernel. It is defined as $K_{hybrid} = \theta K_{path} + (1-\theta)K_{cs}$ $(0 \leq \theta \leq 1)$ (for the detailed definitions of K_{path} and K_{cs}, please refer to Che et al. (2006)). Here, we use our grammar-driven tree kernel to compute K_{path} and K_{cs}, and we call it grammar-driven hybrid tree kernel while Che et al. (2006)'s is non-grammar-driven hybrid convolution tree kernel.

We use a greedy strategy to fine-tune parameters. Evaluation on the development set shows that our kernel yields the best performance when λ (decay factor of tree kernel), λ_1 and λ_2 (two penalty factors for the grammar-driven kernel), θ (hybrid kernel parameter) and c (a SVM training parameter to balance training error and margin) are set to 0.4, 0.6, 0.3, 0.6 and 2.4, respectively. For other parameters, we use default setting. In the CoNLL 2005 benchmark data, we get 647 rules with optional nodes out of the total 6,534 grammar rules and define three equivalent node feature sets as below:

- JJ, JJR, JJS
- RB, RBR, RBS
- NN, NNS, NNP, NNPS, NAC, NX

Here, the verb feature set "VB, VBD, VBG, VBN, VBP, VBZ" is removed since the voice information is very indicative to the arguments of ARG0 (Agent, operator) and ARG1 (Thing operated).

Methods	Accuracy (%)
Baseline: Non-grammar-driven	85.21
+Approximate Node Matching	86.27
+Approximate Substructure Matching	87.12
Ours: Grammar-driven Substructure and Node Matching	87.96
Feature-based method with polynomial kernel (d = 2)	89.92

Table 2: Performance comparison

4.2 Experimental Results

Table 2 compares the performances of different methods on the test set. First, we can see that the new grammar-driven hybrid convolution tree kernel significantly outperforms (χ^2 test with p=0.05) the non-grammar one with an absolute improvement of 2.75 (87.96-85.21) percentage, representing a relative error rate reduction of 18.6% (2.75/(100-85.21)) . It suggests that 1) the linguistically motivated structure features are very useful for semantic role classification and 2) the grammar-driven kernel is much more effective in capturing such kinds of features due to the consideration of linguistic knowledge. Moreover, Table 2 shows that 1) both the grammar-driven approximate node matching and the grammar-driven approximate substructure matching are very useful in modeling syntactic tree structures for SRL since they contribute relative error rate reduction of 7.2% ((86.27-85.21)/(100-85.21)) and 12.9% ((87.12-85.21)/(100-85.21)), respectively; 2) the grammar-driven approximate substructure matching is more effective than the grammar-driven approximate node matching. However, we find that the performance of the grammar-driven kernel is still a bit lower than the feature-based method. This is not surprising since tree kernel methods only focus on modeling tree structure information. In this paper, it captures the syntactic parse tree structure features only while the features used in the feature-based methods cover more knowledge sources.

In order to make full use of the syntactic structure information and the other useful diverse flat features, we present a composite kernel to combine the grammar-driven hybrid kernel and feature-based method with polynomial kernel:

$$K_{comp} = \gamma K_{hybrid} + (1-\gamma) K_{poly} \qquad (0 \le \gamma \le 1)$$

Evaluation on the development set shows that the composite kernel yields the best performance when γ is set to 0.3. Using the same setting, the system achieves the performance of 91.02% in Accuracy in the same test set. It shows statistically significant improvement (χ^2 test with p= 0.10) over using the standard features with the polynomial kernel ($\gamma = 0$, Accuracy = 89.92%) and using the grammar-driven hybrid convolution tree kernel ($\gamma = 1$, Accuracy = 87.96%). The main reason is that the tree kernel can capture effectively more structure features while the standard flat features can cover some other useful features, such as Voice, SubCat, which are hard to be covered by the tree kernel. The experimental results suggest that these two kinds of methods are complementary to each other.

In order to further compare with other methods, we also do experiments on the dataset of English PropBank I (LDC2004T14). The training, development and test sets follow the conventional split of Sections 02-21, 00 and 23. Table 3 compares our method with other previously best-reported methods with the same setting as discussed previously. It shows that our method outperforms the previous best-reported one with a relative error rate reduction of 10.8% (0.97/(100-91)). This further verifies the effectiveness of the grammar-driven kernel method for semantic role classification.

Method	Accuracy (%)
Ours (Composite Kernel)	**91.97**
Moschitti (2006): PAF kernel only	87.7
Jiang et al. (2005): feature based	90.50
Pradhan et al. (2005a): feature based	91.0

Table 3: Performance comparison between our method and previous work

Method	Training Time	
	4 Sections	19 Sections
Ours: grammar-driven tree kernel	~8.1 hours	~7.9 days
Moschitti (2006): non-grammar-driven tree kernel	~7.9 hours	~7.1 days

Table 4: Training time comparison

Table 4 reports the training times of the two kernels. We can see that 1) the two kinds of convolution tree kernels have similar computing time. Although computing the grammar-driven one requires exponential time in its worst case, however, in practice, it may only need $O(|N_1| \cdot |N_2|)$ or linear and 2) it is very time-consuming to train a SVM classifier in a large dataset.

5 Conclusion and Future Work

In this paper, we propose a novel grammar-driven convolution tree kernel for semantic role classification. More linguistic knowledge is considered in the new kernel design. The experimental results verify that the grammar-driven kernel is more effective in capturing syntactic structure features than the previous convolution tree kernel because it allows grammar-driven approximate matching of substructures and node features. We also discuss the criteria to determine the optional nodes in a

CFG rule in defining our grammar-driven convolution tree kernel.

The extension of our work is to improve the performance of the entire semantic role labeling system using the grammar-driven tree kernel, including all four stages: pruning, semantic role identification, classification and post inference. In addition, a more interesting research topic is to study how to integrate linguistic knowledge and tree kernel methods to do feature selection for tree kernel-based NLP applications (Suzuki et al., 2004). In detail, a linguistics and statistics-based theory that can suggest the effectiveness of different substructure features and whether they should be generated or not by the tree kernels would be worked out.

References

C. F. Baker, C. J. Fillmore, and J. B. Lowe. 1998. *The Berkeley FrameNet Project*. COLING-ACL-1998

Xavier Carreras and Lluıs Màrquez. 2004. Introduction *to the CoNLL-2004 shared task: Semantic role labeling*. CoNLL-2004

Xavier Carreras and Lluıs Màrquez. 2005. Introduction *to the CoNLL-2005 shared task: Semantic role labeling*. CoNLL-2005

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings ofNAACL-2000*

Wanxiang Che, Min Zhang, Ting Liu and Sheng Li. 2006. *A hybrid convolution tree kernel for semantic role labeling*. COLING-ACL-2006(poster)

Michael Collins and Nigel Duffy. 2001. *Convolution kernels for natural language*. NIPS-2001

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

David Haussler. 1999. Convolution kernels on discrete structures. Technical Report UCSC-CRL-99-10

Zheng Ping Jiang, Jia Li and Hwee Tou Ng. 2005. *Semantic argument classification exploiting argument interdependence*. IJCAI-2005

T. Joachims. 1998. *Text Categorization with Support Vecor Machine: learning with many relevant features*. ECML-1998

Kashima H. and Koyanagi T. 2003. *Kernels for Semi-Structured Data*. ICML-2003

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*. ACL-2004

Alessandro Moschitti. 2006. *Syntactic kernels for natural language learning: the semantic role labeling case*. HLT-NAACL-2006 (short paper)

Rodney D. Nielsen and Sameer Pradhan. 2004. *Mixing weak learners in semantic parsing*. 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. *Journal of Machine Learning*

Sameer Pradhan, Wayne Ward, Kadri Hacioglu, James Martin and Daniel Jurafsky. 2005b. *Semantic role labeling using different syntactic views.* ACL-2005

Vasin Punyakanok, Dan Roth, Wen-tau Yih and Dav Zimak. 2004. *Semantic role labeling via integer linear programming inference*. COLING-2004

Vasin Punyakanok, Dan Roth and Wen Tau Yih. 2005. *The necessity of syntactic parsing for semantic role labeling*. IJCAI-2005

Libin Shen, Anoop Sarkar and A. K. Joshi. 2003. *Using LTAG based features in parse reranking*. EMNLP-03

Jun Suzuki, Hideki Isozaki and Eisaku Maede. 2004. *Convolution kernels with feature selection for Natural Language processing tasks*. ACL-2004

Vladimir N. Vapnik. 1998. *Statistical Learning Theory.* Wiley

Nianwen Xue and Martha Palmer. 2004. *Calibrating features for semantic role labeling*. EMNLP-2004

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *Machine Learning Research,* 3:1083–1106

Min Zhang, Jie Zhang, Jian Su and Guodong Zhou. 2006. *A Composite Kernel to Extract Relations between Entities with both Flat and Structured Features*. COLING-ACL-2006