Tree Kernel-based Relation Extraction with Context-Sensitive Structured Parse Tree Information

GuoDong ZHOU[1,2] **Min ZHANG**[2] **Dong Hong JI**[2] **QiaoMing ZHU**[1]

[1]School of Computer Science & Technology
Soochow Univ.
Suzhou, China 215006
Email: {gdzhou,qmzhu}@suda.edu.cn

[2] Institute for Infocomm Research
Heng Mui Keng Terrace
Singapore 119613
Email: {zhougd, mzhang, dhji}@i2r.a-star.edu.sg

Abstract

This paper proposes a tree kernel with context-sensitive structured parse tree information for relation extraction. It resolves two critical problems in previous tree kernels for relation extraction in two ways. First, it automatically determines a dynamic context-sensitive tree span for relation extraction by extending the widely-used Shortest Path-enclosed Tree (SPT) to include necessary context information outside SPT. Second, it proposes a context-sensitive convolution tree kernel, which enumerates both context-free and context-sensitive sub-trees by considering their ancestor node paths as their contexts. Moreover, this paper evaluates the complementary nature between our tree kernel and a state-of-the-art linear kernel. Evaluation on the ACE RDC corpora shows that our dynamic context-sensitive tree span is much more suitable for relation extraction than SPT and our tree kernel outperforms the state-of-the-art Collins and Duffy's convolution tree kernel. It also shows that our tree kernel achieves much better performance than the state-of-the-art linear kernels. Finally, it shows that feature-based and tree kernel-based methods much complement each other and the composite kernel can well integrate both flat and structured features.

1 Introduction

Relation extraction is to find various predefined semantic relations between pairs of entities in text. The research in relation extraction has been promoted by the Message Understanding Conferences (MUCs) (MUC, 1987-1998) and the NIST Automatic Content Extraction (ACE) program (ACE, 2002-2005). According to the ACE Program, an entity is an object or a set of objects in the world and a relation is an explicitly or implicitly stated relationship among entities. For example, the sentence "Bill Gates is the chairman and chief software architect of Microsoft Corporation." conveys the ACE-style relation "EMPLOYMENT.exec" between the entities "Bill Gates" (person name) and "Microsoft Corporation" (organization name). Extraction of semantic relations between entities can be very useful in many applications such as question answering, e.g. to answer the query "Who is the president of the United States?", and information retrieval, e.g. to expand the query "George W. Bush" with "the president of the United States" via his relationship with "the United States".

Many researches have been done in relation extraction. Among them, feature-based methods (Kambhatla 2004; Zhou et al., 2005) achieve certain success by employing a large amount of diverse linguistic features, varying from lexical knowledge, entity-related information to syntactic parse trees, dependency trees and semantic information. However, it is difficult for them to effectively capture structured parse tree information (Zhou et al 2005), which is critical for further performance improvement in relation extraction.

As an alternative to feature-based methods, tree kernel-based methods provide an elegant solution to explore implicitly structured features by directly computing the similarity between two trees. Although earlier researches (Zelenko et al 2003; Culotta and Sorensen 2004; Bunescu and Mooney 2005a) only achieve success on simple tasks and fail on complex tasks, such as the ACE RDC task, tree kernel-based methods achieve much progress recently. As the state-of-the-art, Zhang et al (2006) applied the convolution tree kernel (Collins and Duffy 2001) and achieved comparable performance with a state-of-the-art linear kernel (Zhou et al 2005) on the 5 relation types in the ACE RDC 2003 corpus.

However, there are two problems in Collins and Duffy's convolution tree kernel for relation extraction. The first is that the sub-trees enumerated in the tree kernel computation are context-free. That is, each sub-tree enumerated in the tree kernel computation

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 728–736, Prague, June 2007.

does not consider the context information outside the sub-tree. The second is to decide a proper tree span in relation extraction. Zhang et al (2006) explored five tree spans in relation extraction and it was a bit surprising to find that the Shortest Path-enclosed Tree (SPT, i.e. the sub-tree enclosed by the shortest path linking two involved entities in the parse tree) performed best. This is contrast to our intuition. For example, "got married" is critical to determine the relationship between "John" and "Mary" in the sentence "John and Mary got married…" as shown in Figure 1(e). It is obvious that the information contained in SPT ("John and Marry") is not enough to determine their relationship.

This paper proposes a context-sensitive convolution tree kernel for relation extraction to resolve the above two problems. It first automatically determines a dynamic context-sensitive tree span for relation extraction by extending the Shortest Path-enclosed Tree (SPT) to include necessary context information outside SPT. Then it proposes a context-sensitive convolution tree kernel, which not only enumerates context-free sub-trees but also context-sensitive sub-trees by considering their ancestor node paths as their contexts. Moreover, this paper evaluates the complementary nature of different linear kernels and tree kernels via a composite kernel.

The layout of this paper is as follows. In Section 2, we review related work in more details. Then, the dynamic context-sensitive tree span and the context-sensitive convolution tree kernel are proposed in Section 3 while Section 4 shows the experimental results. Finally, we conclude our work in Section 5.

2 Related Work

The relation extraction task was first introduced as part of the Template Element task in MUC6 and then formulated as the Template Relation task in MUC7. Since then, many methods, such as feature-based (Kambhatla 2004; Zhou et al 2005, 2006), tree kernel-based (Zelenko et al 2003; Culotta and Sorensen 2004; Bunescu and Mooney 2005a; Zhang et al 2006) and composite kernel-based (Zhao and Grishman 2005; Zhang et al 2006), have been proposed in literature.

For the feature-based methods, Kambhatla (2004) employed Maximum Entropy models to combine diverse lexical, syntactic and semantic features in relation extraction, and achieved the F-measure of 52.8 on the 24 relation subtypes in the ACE RDC 2003 corpus. Zhou et al (2005) further systematically explored diverse features through a linear kernel and Support Vector Machines, and achieved the F-measures of 68.0 and 55.5 on the 5 relation types and the 24 relation subtypes in the ACE RDC 2003 corpus respectively. One problem with the feature-based methods is that they need extensive feature engineering. Another problem is that, although they can explore some structured information in the parse tree (e.g. Kambhatla (2004) used the non-terminal path connecting the given two entities in a parse tree while Zhou et al. (2005) introduced additional chunking features to enhance the performance), it is found difficult to well preserve structured information in the parse trees using the feature-based methods. Zhou et al (2006) further improved the performance by exploring the commonality among related classes in a class hierarchy using hierarchical learning strategy.

As an alternative to the feature-based methods, the kernel-based methods (Haussler, 1999) have been proposed to implicitly explore various features in a high dimensional space by employing a kernel to calculate the similarity between two objects directly. In particular, the kernel-based methods could be very effective at reducing the burden of feature engineering for structured objects in NLP researches, e.g. the tree structure in relation extraction.

Zelenko et al. (2003) proposed a kernel between two parse trees, which recursively matches nodes from roots to leaves in a top-down manner. For each pair of matched nodes, a subsequence kernel on their child nodes is invoked. They achieved quite success on two simple relation extraction tasks. Culotta and Sorensen (2004) extended this work to estimate similarity between augmented dependency trees and achieved the F-measure of 45.8 on the 5 relation types in the ACE RDC 2003 corpus. One problem with the above two tree kernels is that matched nodes must be at the same height and have the same path to the root node. Bunescu and Mooney (2005a) proposed a shortest path dependency tree kernel, which just sums up the number of common word classes at each position in the two paths, and achieved the F-measure of 52.5 on the 5 relation types in the ACE RDC 2003 corpus. They argued that the information to model a relationship between two entities can be typically captured by the shortest path between them in the dependency graph. While the shortest path may not be able to well preserve structured dependency tree information, another problem with their kernel is that the two paths should have same length. This makes it suffer from the similar behavior with that of Culotta and Sorensen (2004): high precision but very low recall.

As the state-of-the-art tree kernel-based method, Zhang et al (2006) explored various structured feature spaces and used the convolution tree kernel over parse trees (Collins and Duffy 2001) to model syntactic structured information for relation extraction. They achieved the F-measures of 61.9 and 63.6 on the 5 relation types of the ACE RDC 2003 corpus and the 7 relation types of the ACE RDC 2004 corpus respectively without entity-related information while the F-measure on the 5 relation types in the ACE RDC 2003 corpus reached 68.7 when entity-related information was included in the parse tree. One problem with Collins and Duffy's convolution tree kernel is that the sub-trees involved in the tree kernel computation are context-free, that is, they do not consider the information outside the sub-trees. This is different from the tree kernel in Culota and Sorensen (2004), where the sub-trees involved in the tree kernel computation are context-sensitive (that is, with the path from the tree root node to the sub-tree root node in consideration). Zhang et al (2006) also showed that the widely-used Shortest Path-enclosed Tree (SPT) performed best. One problem with SPT is that it fails to capture the contextual information outside the shortest path, which is important for relation extraction in many cases. Our random selection of 100 positive training instances from the ACE RDC 2003 training corpus shows that ~25% of the cases need contextual information outside the shortest path. Among other kernels, Bunescu and Mooney (2005b) proposed a subsequence kernel and applied it in protein interaction and ACE relation extraction tasks.

In order to integrate the advantages of feature-based and tree kernel-based methods, some researchers have turned to composite kernel-based methods. Zhao and Grishman (2005) defined several feature-based composite kernels to integrate diverse features for relation extraction and achieved the F-measure of 70.4 on the 7 relation types of the ACE RDC 2004 corpus. Zhang et al (2006) proposed two composite kernels to integrate a linear kernel and Collins and Duffy's convolution tree kernel. It achieved the F-measure of 70.9/57.2 on the 5 relation types/24 relation subtypes in the ACE RDC 2003 corpus and the F-measure of 72.1/63.6 on the 7 relation types/23 relation subtypes in the ACE RDC 2004 corpus.

The above discussion suggests that structured information in the parse tree may not be fully utilized in the previous works, regardless of feature-based, tree kernel-based or composite kernel-based methods. Compared with the previous works, this paper proposes a dynamic context-sensitive tree span trying to cover necessary structured information and a context-sensitive convolution tree kernel considering both context-free and context-sensitive sub-trees. Furthermore, a composite kernel is applied to combine our tree kernel and a state-of-the-art linear kernel for integrating both flat and structured features in relation extraction as well as validating their complementary nature.

3 Context Sensitive Convolution Tree Kernel for Relation Extraction

In this section, we first propose an algorithm to dynamically determine a proper context-sensitive tree span and then a context-sensitive convolution tree kernel for relation extraction.

3.1 Dynamic Context-Sensitive Tree Span in Relation Extraction

A relation instance between two entities is encapsulated by a parse tree. Thus, it is critical to understand which portion of a parse tree is important in the tree kernel calculation. Zhang et al (2006) systematically explored seven different tree spans, including the Shortest Path-enclosed Tree (SPT) and a Context-Sensitive Path-enclosed Tree[1] (CSPT), and found that SPT performed best. That is, SPT even outperforms CSPT. This is contrary to our intuition. For example, "got married" is critical to determine the relationship between "John" and "Mary" in the sentence "John and Mary got married…" as shown in Figure 1(e), and the information contained in SPT ("John and Mary") is not enough to determine their relationship. Obviously, context-sensitive tree spans should have the potential for better performance. One problem with the context-sensitive tree span explored in Zhang et al (2006) is that it only considers the availability of entities' siblings and fails to consider following two factors:

1) Whether is the information contained in SPT enough to determine the relationship between two entities? It depends. In the embedded cases, SPT is enough. For example, "John's wife" is enough to determine the relationship between "John" and "John's wife" in the sentence "John's wife got a good job… " as shown in Figure 1(a). However, SPT is not enough in the coordinated cases, e.g. to determine the relationship between "John" and "Mary" in the sentence "John and Mary got married…" as shown in Figure 1(e).

[1] CSPT means SPT extending with the 1st left sibling of the node of entity 1 and the 1st right sibling of the node of entity 2. In the case of no available sibling, it moves to the parent of current node and repeat the same process until a sibling is available or the root is reached.

2) How can we extend SPT to include necessary context information if there is no enough information in SPT for relation extraction?

To answer the above two questions, we randomly chose 100 positive instances from the ACE RDC 2003 training data and studied their necessary tree spans. It was observed that we can classify them into 5 categories: 1) embedded (37 instances), where one entity is embedded in another entity, e.g. “John” and “John’s wife”as shown in Figure 1(a); 2) PP-linked (21 instances), where one entity is linked to another entity via PP attachment, e.g. “CEO” and “Microsoft” in the sentence “CEO of Microsoft announced … ”as shown in Figure 1(b); 3) semi-structured (15 instances), where the sentence consists of a sequence of noun phrases (including the two given entities), e.g. “Jane” and “ABC news” in the sentence “Jane, ABC news, California.” as shown in Figure 1(c); 4) descriptive (7 instances), e.g. the citizenship between “his mother” and “Lebanese” in the sentence “his mother Lebanese landed at …” as shown in Figure 1(d); 5) predicate-linked and others (19 instances, including coordinated cases), where the predicate information is necessary to determine the relationship between two entities, e.g. “John” and “Mary” in the sentence “John and Mary got married…” as shown in Figure 1(e);

Based on the above observations, we implement an algorithm to determine the necessary tree span for the relation extract task. The idea behind the algorithm is that the necessary tree span for a relation should be determined dynamically according to its tree span category and context. Given a parsed tree and two entities in consideration, it first determines the tree span category and then extends the tree span accordingly. By default, we adopt the Shortest Path-enclosed Tree (SPT) as our tree span. We only expand the tree span when the tree span belongs to the “predicate-linked” category. This is based on our observation that the tree spans belonging to the “predicate-linked” category vary much syntactically and majority (~70%) of them need information outside SPT while it is quite safe (>90%) to use SPT as the tree span for the remaining categories. In our algorithm, the expansion is done by first moving up until a predicate-headed phrase is found and then moving down along the predicated-headed path to the predicate terminal node. Figure 1(e) shows an example for the “predicate-linked” category where the lines with arrows indicate the expansion path.

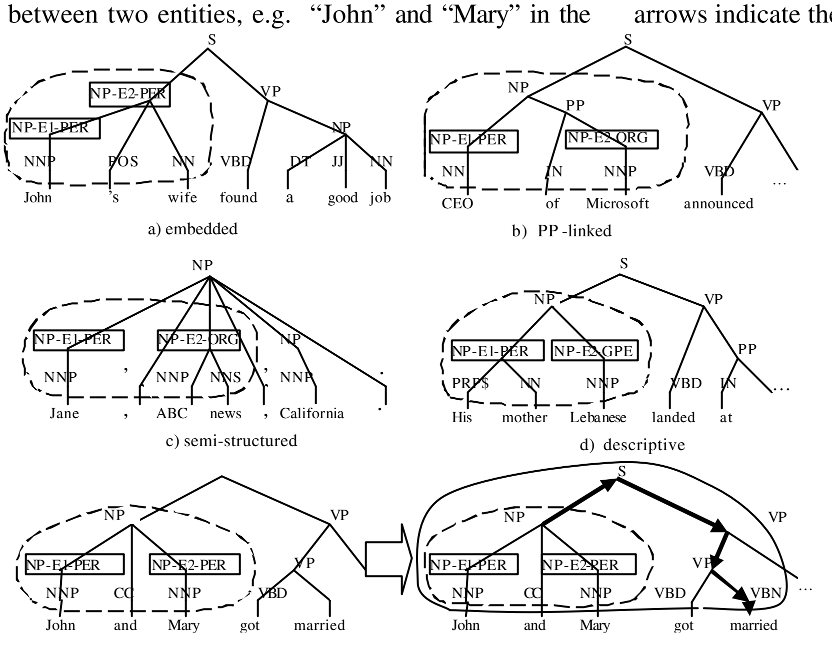

Figure 1: Different tree span categories with SPT (dotted circle) and an example of the dynamic context-sensitive tree span (solid circle)

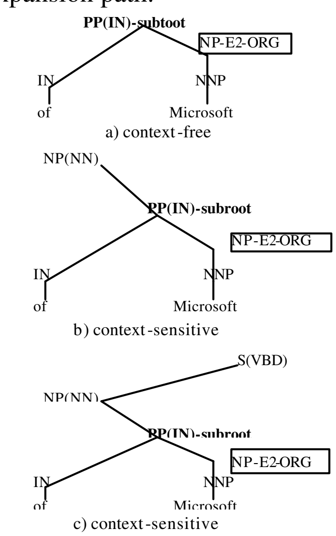

Figure 2: Examples of context-free and context-sensitive sub-trees related with Figure 1(b). Note: the bold node is the root for a sub-tree.

A problem with our algorithm is how to determine whether an entity pair belongs to the “predicate-linked” category. In this paper, a simple method is applied by regarding the “predicate-linked” category as the default category. That is, those entity pairs, which do not belong to the four well defined and easily detected categories (i.e. embedded, PP-liked, semi-structured and descriptive), are classified into the “predicate-linked” category.

Since "predicate-linked" instances only occupy ~20% of cases, this explains why SPT performs better than the Context-Sensitive Path-enclosed Tree (CSPT) as described in Zhang et al (2006): consistently adopting CSPT may introduce too much noise/unnecessary information in the tree kernel.

3.2 Context-Sensitive Convolution Tree Kernel

Given any tree span, e.g. the dynamic context-sensitive tree span in the last subsection, we now study how to measure the similarity between two trees, using a convolution tree kernel. A convolution kernel (Haussler D., 1999) aims to capture structured information in terms of substructures. As a specialized convolution kernel, Collins and Duffy's convolution tree kernel $K_C(T_1,T_2)$ ('C' for convolution) counts the number of common sub-trees (sub-structures) as the syntactic structure similarity between two parse trees T_1 and T_2 (Collins and Duffy 2001):

$$K_C(T_1,T_2)=\sum_{n_1\in N_1, n_2\in N_2}\Delta(n_1,n_2) \qquad (1)$$

where N_j is the set of nodes in tree T_j, and $\Delta(n_1,n_2)$ evaluates the common sub-trees rooted at n_1 and n_2 [2] and is computed recursively as follows:

1) If the context-free productions (Context-Free Grammar(CFG) rules) at n_1 and n_2 are different, $\Delta(n_1,n_2)=0$; Otherwise go to 2.
2) If both n_1 and n_2 are POS tags, $\Delta(n_1,n_2)=1\times\lambda$; Otherwise go to 3.
3) Calculate $\Delta(n_1,n_2)$ recursively as:

$$\Delta(n_1,n_2)=\lambda\prod_{k=1}^{\#ch(n_1)}(1+\Delta(ch(n_1,k),ch(n_2,k)) \qquad (2)$$

where $\#ch(n)$ is the number of children of node n, $ch(n,k)$ is the k^{th} child of node n and λ ($0<\lambda<1$) is the decay factor in order to make the kernel value less variable with respect to different sub-tree sizes.

This convolution tree kernel has been successfully applied by Zhang et al (2006) in relation extraction. However, there is one problem with this tree kernel: the sub-trees involved in the tree kernel computation are context-free (That is, they do not consider the information outside the sub-trees). This is contrast to the tree kernel proposed in Culota and Sorensen (2004) which is context-sensitive, that is, it considers the path from the tree root node to the sub-tree root node. In order to integrate the advantages of both tree kernels and resolve the problem in Collins and Duffy's convolution tree kernel, this paper proposes a context-sensitive convolution tree kernel. It works by taking ancestral information (i.e. the root node path) of sub-trees into consideration:

$$K_C(T[1],T[2])=\sum_{i=1}^{m}\sum_{n_1^i[1]\in N_1^i[1], n_1^i[2]\in N_1^i[2]}\Delta(n_1^i[1],n_1^i[2]) \qquad (3)$$

Where

- $N_1^i[j]$ is the set of root node paths with length i in tree *T[j]* while the maximal length of a root node path is defined by *m*.
- $n_1^i[j]=(n_1n_2...n_i)[j]$ is a root node path with length i in tree *T[j]*, which takes into account the i-1 ancestral nodes $n_2^i[j]$ of $n_1[j]$ in *T[j]*. Here, $n_{k+1}[j]$ is the parent of $n_k[j]$ and $n_1[j]$ is the root node of a context-free sub-tree in *T[j]*. For better differentiation, the label of each ancestral node in $n_1^i[j]$ is augmented with the POS tag of its head word.
- $\Delta(n_1^i[1],n_1^i[2])$ measures the common context-sensitive sub-trees rooted at root node paths $n_1^i[1]$ and $n_1^i[2]$ [3]. In our tree kernel, a sub-tree becomes context-sensitive with its dependence on the root node path instead of the root node itself. Figure 2 shows a few examples of context-sensitive sub-trees with comparison to context-free sub-trees.

Similar to Collins and Duffy (2001), our tree kernel computes $\Delta(n_1^i[1],n_1^i[2])$ recursively as follows:

1) If the context-sensitive productions (Context-Sensitive Grammar (CSG) rules with root node paths as their left hand sides) rooted at $n_1^i[1]$ and $n_1^i[2]$ are different, return $\Delta(n_1^i[1],n_1^i[2])=0$; Otherwise go to Step 2.
2) If both $n_1[1]$ and $n_1[2]$ are POS tags, $\Delta(n_1^i[1],n_1^i[2])=\lambda$; Otherwise go to Step 3.

[2] That is, each node n encodes the identity of a sub-tree rooted at n and, if there are two nodes in the tree with the same label, the summation will go over both of them.

[3] That is, each root node path n_1^i encodes the identity of a context-sensitive sub-tree rooted at n_1^i and, if there are two root node paths in the tree with the same label sequence, the summation will go over both of them.

3) Calculate $\Delta(n_1^i[1], n_1^i[2])$ recursively as:

$$\Delta(n_1^i[1], n_1^i[2]) = \lambda \prod_{k=1}^{\#ch(n_1^i[1])} (1 + \Delta(ch(n_1^i[1], k), ch(n_1^i[2], k)) \quad (4)$$

where $ch(n_1^i[j], k)$ is the k^{th} context-sensitive child of the context-sensitive sub-tree rooted at $n_1^i[j]$ with $\#ch(n_1^i[j])$ the number of the context-sensitive children. Here, λ $(0 < \lambda < 1)$ is the decay factor in order to make the kernel value less variable with respect to different sizes of the context-sensitive sub-trees.

It is worth comparing our tree kernel with previous tree kernels. Obviously, our tree kernel is an extension of Collins and Duffy's convolution tree kernel, which is a special case of our tree kernel (if *m=1* in Equation (3)). Our tree kernel not only counts the occurrence of each context-free sub-tree, which does not consider its ancestors, but also counts the occurrence of each context-sensitive sub-tree, which considers its ancestors. As a result, our tree kernel is not limited by the constraints in previous tree kernels (as discussed in Section 2), such as Collins and Duffy (2001), Zhang et al (2006), Culotta and Sorensen (2004) and Bunescu and Mooney (2005a). Finally, let's study the computational issue with our tree kernel. Although our tree kernel takes the context-sensitive sub-trees into consideration, it only slightly increases the computational burden, compared with Collins and Duffy's convolution tree kernel. This is due to that $\Delta(n_1[1], n_1[2]) = 0$ holds for the majority of context-free sub-tree pairs (Collins and Duffy 2001) and that computation for context-sensitive sub-tree pairs is necessary only when $\Delta(n_1[1], n_1[2]) \neq 0$ and the context-sensitive sub-tree pairs have the same root node path(i.e. $n_1^i[1] = n_1^i[2]$ in Equation (3)).

4 Experimentation

This paper uses the ACE RDC 2003 and 2004 corpora provided by LDC in all our experiments.

4.1 Experimental Setting

The ACE RDC corpora are gathered from various newspapers, newswire and broadcasts. In the 2003 corpus, the training set consists of 674 documents and 9683 positive relation instances while the test set consists of 97 documents and 1386 positive relation instances. The 2003 corpus defines **5** entity types, **5** major relation types and **24** relation subtypes. All the reported performances in this paper on the ACE RDC 2003 corpus are evaluated on the test data. The 2004 corpus contains 451 documents and 5702 positive relation instances. It redefines **7** entity types, **7** major relation types and **23** relation subtypes. For comparison, we use the same setting as Zhang et al (2006) by applying a 5-fold cross-validation on a subset of the 2004 data, containing 348 documents and 4400 relation instances. That is, all the reported performances in this paper on the ACE RDC 2004 corpus are evaluated using 5-fold cross validation on the entire corpus.

Both corpora are parsed using Charniak's parser (Charniak, 2001) with the boundaries of all the entity mentions kept[4]. We iterate over all pairs of entity mentions occurring in the same sentence to generate potential relation instances[5]. In our experimentation, SVM (SVMLight, Joachims(1998)) is selected as our classifier. For efficiency, we apply the *one vs. others* strategy, which builds K classifiers so as to separate one class from all others. The training parameters are chosen using cross-validation on the ACE RDC 2003 training data. In particular, λ in our tree kernel is fine-tuned to 0.5. This suggests that about 50% discount is done as our tree kernel moves down one level in computing $\Delta(n_1^i[1], n_1^i[2])$.

4.2 Experimental Results

First, we systematically evaluate the context-sensitive convolution tree kernel and the dynamic context-sensitive tree span proposed in this paper.

Then, we evaluate the complementary nature between our tree kernel and a state-of-the-art linear kernel via a composite kernel. Generally different feature-based methods and tree kernel-based methods have their own merits. It is usually easy to build a system using a feature-based method and achieve the state-of-the-art performance, while tree kernel-based methods hold the potential for further performance improvement. Therefore, it is always a good idea to integrate them via a composite kernel.

[4] This can be done by first representing all entity mentions with their head words and then restoring all the entity mentions after parsing. Moreover, please note that the final performance of relation extraction may change much with different range of parsing errors. We will study this issue in the near future.

[5] In this paper, we only measure the performance of relation extraction on "true" mentions with "true" chaining of co-reference (i.e. as annotated by LDC annotators). Moreover, we only model explicit relations and explicitly model the argument order of the two mentions involved.

Finally, we compare our system with the state-of-the-art systems in the literature.

Context-Sensitive Convolution Tree Kernel

In this paper, the m parameter of our context-sensitive convolution tree kernel as shown in Equation (3) indicates the maximal length of root node paths and is optimized to 3 using 5-fold cross validation on the ACE RDC 2003 training data. Table 1 compares the impact of different m in context-sensitive convolution tree kernels using the Shortest Path-enclosed Tree (SPT) (as described in Zhang et al (2006)) on the major relation types of the ACE RDC 2003 and 2004 corpora, in details. It also shows that our tree kernel achieves best performance on the test data using SPT with $m = 3$, which outperforms the one with $m = 1$ by ~2.3 in F-measure. This suggests the parent and grandparent nodes of a sub-tree contains much information for relation extraction while considering more ancestral nodes may not help. This may be due to that, although our experimentation on the training data indicates that more than 80% (on average) of subtrees has a root node path longer than 3 (since most of the subtrees are deep from the root node and more than 90% of the parsed trees in the training data are deeper than 6 levels), including a root node path longer than 3 may be vulnerable to the full parsing errors and have negative impact. Table 1 also evaluates the impact of entity-related information in our tree kernel by attaching entity type information (e.g. "PER" in the entity node 1 of Figure 1(b)) into both entity nodes. It shows that such information can significantly improve the performance by ~6.0 in F-measure. In all the following experiments, we will apply our tree kernel with m=3 and entity-related information by default.

Table 2 compares the dynamic context-sensitive tree span with SPT using our tree kernel. It shows that the dynamic tree span can futher improve the performance by ~1.2 in F-measure[6]. This suggests the usefulness of extending the tree span beyond SPT for the "predicate-linked" tree span category. In the future work, we will further explore expanding the dynamic tree span beyond SPT for the remaining tree span categories.

[6] Significance test shows that the dynamic tree span performs statistically significantly better than SPT with p-values smaller than 0.05.

m	P(%)	R(%)	F
1	72.3(72.7)	56.6(53.8)	63.5(61.8)
2	74.9(75.2)	57.9(54.7)	65.3(63.5)
3	75.7(76.1)	58.3(55.1)	65.9(64.0)
4	76.0(75.9)	58.3(55.3)	66.0(63.9)

a) without entity-related information

m	P(%)	R(%)	F
1	77.2(76.9)	63.5(60.8)	69.7(67.9)
2	79.1(78.6)	65.0(62.2)	71.3(69.4)
3	79.6(79.4)	65.6(62.5)	71.9(69.9)
4	79.4(79.1)	65.6(62.3)	71.8(69.7)

b) with entity-related information

Table 1: Evaluation of context-sensitive convolution tree kernels using SPT on the major relation types of the ACE RDC 2003 (inside the parentheses) and 2004 (outside the parentheses) corpora.

Tree Span	P(%)	R(%)	F
Shortest Path-enclosed Tree	79.6 (79.4)	65.6 (62.5)	71.9 (69.9)
Dynamic Context-Sensitive Tee	81.1 (80.1)	66.7 (63.8)	73.2 (71.0)

Table 2: Comparison of dynamic context-sensitive tree span with SPT using our context-sensitive convolution tree kernel on the major relation types of the ACE RDC 2003 (inside the parentheses) and 2004 (outside the parentheses) corpora. 18% of positive instances in the ACE RDC 2003 test data belong to the predicate-linked category.

Composite Kernel

In this paper, a composite kernel via polynomial interpolation, as described Zhang et al (2006), is applied to integrate the proposed context-sensitive convolution tree kernel with a state-of-the-art linear kernel (Zhou et al 2005)[7]:

$$K_1(\bullet,\bullet) = \alpha \cdot K_L^P(\bullet,\bullet) + (1-\alpha) \cdot K_C(\bullet,\bullet) \quad (5)$$

Here, $K_L(\bullet,\bullet)$ and $K_C(\bullet,\bullet)$ indicates the normalized linear kernel and context-sensitive convolution tree kernel respectively while $K^p(\cdot,\cdot)$ is the polynomial expansion of $K(\cdot,\cdot)$ with degree d=2, i.e. $K^p(\cdot,\cdot) = (K(\cdot,\cdot)+1)^2$ and α is the coefficient (α is set to 0.3 using cross-validation).

[7] Here, we use the same set of flat features (i.e. word, entity type, mention level, overlap, base phrase chunking, dependency tree, parse tree and semantic information) as Zhou et al (2005).

Table 3 evaluates the performance of the composite kernel. It shows that the composite kernel much further improves the performance beyond that of either the state-of-the-art linear kernel or our tree kernel and achieves the F-measures of 74.1 and 75.8 on the major relation types of the ACE RDC 2003 and 2004 corpora respectively. This suggests that our tree kernel and the state-of-the-art linear kernel are quite complementary, and that our composite kernel can effectively integrate both flat and structured features.

System	P(%)	R(%)	F
Linear Kernel	78.2 (77.2)	63.4 (60.7)	70.1 (68.0)
Context-Sensitive Convolution Tree Kernel	81.1 (80.1)	66.7 (63.8)	73.2 (71.0)
Composite Kernel	82.2 (80.8)	70.2 (68.4)	75.8 (74.1)

Table 3: Performance of the composite kernel via polynomial interpolation on the major relation types of the ACE RDC 2003 (inside the parentheses) and 2004 (outside the parentheses) corpora

Comparison with Other Systems

ACE RDC 2003	P(%)	R(%)	F
Ours: composite kernel	80.8 (65.2)	68.4 (54.9)	74.1 (59.6)
Zhang et al (2006): composite kernel	77.3 (64.9)	65.6 (51.2)	70.9 (57.2)
Ours: context-sensitive convolution tree kernel	80.1 (63.4)	63.8 (51.9)	71.0 (57.1)
Zhang et al (2006): convolution tree kernel	76.1 (62.4)	62.6 (48.5)	68.7 (54.6)
Bunescu et al (2005): shortest path dependency kernel	65.5 (-)	43.8 (-)	52.5 (-)
Culotta et al (2004): dependency kernel	67.1 (-)	35.0 (-)	45.8 (-)
Zhou et al. (2005): feature-based	77.2 (63.1)	60.7 (49.5)	68.0 (55.5)
Kambhatla (2004): feature-based	- (63.5)	- (45.2)	- (52.8)

Table 4: Comparison of difference systems on the ACE RDC 2003 corpus over both 5 types (outside the parentheses) and 24 subtypes (inside the parentheses)

ACE RDC 2004	P(%)	R(%)	F
Ours: composite kernel	82.2 (70.3)	70.2 (62.2)	75.8 (66.0)
Zhang et al (2006): composite kernel	76.1 (68.6)	68.4 (59.3)	72.1 (63.6)
Zhao et al (2005):[8] composite kernel	69.2 (-)	70.5 (-)	70.4 (-)
Ours: context-sensitive convolution tree kernel	81.1 (68.8)	66.7 (60.3)	73.2 (64.3)
Zhang et al (2006): convolution tree kernel	72.5 (-)	56.7 (-)	63.6 (-)

Table 5: Comparison of difference systems on the ACE RDC 2004 corpus over both 7 types (outside the parentheses) and 23 subtypes (inside the parentheses)

Finally, Tables 4 and 5 compare our system with other state-of-the-art systems[9] on the ACE RDC 2003 and 2004 corpora, respectively. They show that our tree kernel-based system outperforms previous tree kernel-based systems. This is largely due to the context-sensitive nature of our tree kernel which resolves the limitations of the previous tree kernels. They also show that our tree kernel-based system outperforms the state-of-the-art feature-based system. This proves the great potential inherent in the parse tree structure for relation extraction and our tree kernel takes a big stride towards the right direction. Finally, they also show that our composite kernel-based system outperforms other composite kernel-based systems.

5 Conclusion

Structured parse tree information holds great potential for relation extraction. This paper proposes a context-sensitive convolution tree kernel to resolve two critical problems in previous tree kernels for relation extraction by first automatically determining a dynamic context-sensitive tree span and then applying a context-sensitive convolution tree kernel. Moreover, this paper evaluates the complementary nature between our tree kernel and a state-of-the-art linear kernel. Evaluation on the ACE RDC corpora shows that our dynamic context-sensitive tree span is much more suitable for relation extraction than the widely-used Shortest Path-enclosed Tree and our tree kernel outperforms the state-of-the-art Collins and Duffy's convolution tree kernel. It also shows that feature-based

[8] There might be some typing errors for the performance reported in Zhao and Grishman(2005) since P, R and F do not match.

[9] All the state-of-the-art systems apply the entity-related information. It is not supervising: our experiments show that using the entity-related information gives a large performance improvement.

and tree kernel-based methods well complement each other and the composite kernel can effectively integrate both flat and structured features.

To our knowledge, this is the first research to demonstrate that, without extensive feature engineering, an individual tree kernel can achieve much better performance than the state-of-the-art linear kernel in relation extraction. This shows the great potential of structured parse tree information for relation extraction and our tree kernel takes a big stride towards the right direction.

For the future work, we will focus on improving the context-sensitive convolution tree kernel by exploring more useful context information. Moreover, we will explore more entity-related information in the parse tree. Our preliminary work of including the entity type information significantly improves the performance. Finally, we will study how to resolve the data imbalance and sparseness issues from the learning algorithm viewpoint.

Acknowledgement

This research is supported by Project 60673041 under the National Natural Science Foundation of China and Project 2006AA01Z147 under the "863" National High-Tech Research and Development of China. We would also like to thank the critical and insightful comments from the four anonymous reviewers.

References

ACE. (2000-2005). Automatic Content Extraction. http://www.ldc.upenn.edu/Projects/ACE/

Bunescu R. & Mooney R.J. (2005a). A shortest path dependency kernel for relation extraction. *HLT/EMNLP'2005*: 724-731. 6-8 Oct 2005. Vancover, B.C.

Bunescu R. & Mooney R.J. (2005b). Subsequence Kernels for Relation Extraction *NIPS'2005*. Vancouver, BC, December 2005

Charniak E. (2001). Immediate-head Parsing for Language Models. *ACL'2001*: 129-137. Toulouse, France

Collins M. and Duffy N. (2001). Convolution Kernels for Natural Language. *NIPS'2001*: 625-632. Cambridge, MA

Culotta A. and Sorensen J. (2004). Dependency tree kernels for relation extraction. *ACL'2004*. 423-429. 21-26 July 2004. Barcelona, Spain.

Haussler D. (1999). Convolution Kernels on Discrete Structures. *Technical Report UCS-CRL-99-10*, University of California, Santa Cruz

Joachims T. (1998). Text Categorization with Support Vector Machine: learning with many relevant features. *ECML-1998*: 137-142. Chemnitz, Germany

Kambhatla N. (2004). Combining lexical, syntactic and semantic features with Maximum Entropy models for extracting relations. *ACL'2004*(Poster). 178-181. 21-26 July 2004. Barcelona, Spain.

MUC. (1987-1998). The NIST MUC website: http://www.itl.nist.gov/iaui/894.02/related_projects/muc/

Zelenko D., Aone C. and Richardella. (2003). Kernel methods for relation extraction. *Journal of Machine Learning Research*. 3(Feb):1083-1106.

Zhang M., Zhang J., Su J. and Zhou GD. (2006). A Composite Kernel to Extract Relations between Entities with both Flat and Structured Features. *COLING-ACL-2006*: 825-832. Sydney, Australia

Zhao S.B. and Grishman R. (2005). Extracting relations with integrated information using kernel methods. *ACL'2005*: 419-426. Univ of Michigan-Ann Arbor, USA, 25-30 June 2005.

Zhou G.D., Su J. Zhang J. and Zhang M. (2005). Exploring various knowledge in relation extraction. *ACL'2005*. 427-434. 25-30 June, Ann Arbor, Michgan, USA.

Zhou G.D., Su J. and Zhang M. (2006). Modeling commonality among related classes in relation extraction, *COLING-ACL'2006*: 121-128. Sydney, Australia.