Information Processing and Management 47 (2011) 349–362

Contents lists available at ScienceDirect

Information Processing and Management

journal homepage: www.elsevier.com/locate/infoproman

Tree kernel-based semantic role labeling with enriched parse tree structure

Zhou GuoDong *, Li Junhui, Fan Jianxi, Zhu Qiaoming

School of Computer Science and Technology, Soochow Univ., 1 ShiZi Street, Suzhou 215006, China

ARTICLE INFO

Article history:
Received 24 December 2008
Received in revised form 1 June 2010
Accepted 16 August 2010
Available online 13 October 2010

Keywords:
Semantic role labeling
Enriched parse tree structure
Convolution tree kernel
Context-sensitiveness
Approximate matching

ABSTRACT

Shallow semantic parsing assigns a simple structure (such as WHO did WHAT to WHOM, WHEN, WHERE, WHY, and HOW) to each predicate in a sentence. It plays a critical role in event-based information extraction and thus is important for deep information processing and management. This paper proposes a tree kernel method for a particular shallow semantic parsing task, called semantic role labeling (SRL), with an enriched parse tree structure. First, a new tree kernel is presented to effectively capture the inherent structured knowledge in a parse tree by enabling the standard convolution tree kernel with context-sensitiveness via considering ancestral information of substructures and approximate matching via allowing insertion/deletion/substitution of tree nodes in the substructures. Second, an enriched parse tree structure is proposed to both well preserve the necessary structured information and effectively avoid noise by differentiating various portions of the parse tree structure. Evaluation on the CoNLL'2005 shared task shows that both the new tree kernel and the enriched parse tree structure contribute much in SRL and our tree kernel method significantly outperforms the state-of-the-art tree kernel methods. Moreover, our tree kernel method is proven rather complementary to the state-of-the-art feature-based methods in that it can better capture structural parse tree information.

1. Introduction

Semantic parsing maps a natural language sentence into a formal representation of its meaning. Due to the difficulty in deep semantic parsing, most of previous work focus on shallow semantic parsing, which assigns a simple structure (such as WHO did WHAT to WHOM, WHEN, WHERE, WHY, and HOW) to each predicate in a sentence. In particular, the well-defined semantic role labeling (SRL) task has been drawing more and more attention in recent years, benefiting from the two well-known projects, FrameNet (Baker, Fillmore, & Lowe, 1998) and PropBank (Palmer, Gildea, & Kingsbury, 2005).

Given a sentence and a predicate (either a target verb or a noun) in it, SRL recognizes and maps all the constituents in the sentence into their corresponding semantic arguments (roles) of the predicate or non-argument. In FrameNet (Baker et al., 1998), rich semantic structures called semantic frames are used to describe predicate-argument structures. These semantic frames schematically represent different scenarios involving various participants, properties and roles, which are called frame elements. Accordingly, all predicates in the same semantic frame share one set of frame elements. For example, predicates "buy" and "sell" belong to the semantic frame named "commercial transaction", which contains frame elements such as "buyer", "seller" and "goods". In PropBank (Palmer et al., 2005), there is much less emphasis on the definition of semantics of the class that the predicates are associated with. Instead, PropBank defines a set of numbered semantic roles (i.e., A0–A5 and AA) to represent core arguments. It is worthwhile to note that predicates vary on the number of core arguments they take. According to Palmer et al. (2005), A0 and A1 generally denote the core arguments exhibiting features of a prototypical

* Corresponding author. Tel.: +86 13402679766; fax: +86 512 65241071.
E-mail addresses: gdzhou@suda.edu.cn (G. Zhou), lijunhui@suda.edu.cn (J. Li), jxfan@suda.edu.cn (J. Fan), qmzhu@suda.edu.cn (Q. Zhu).

0306-4573/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.ipm.2010.08.005

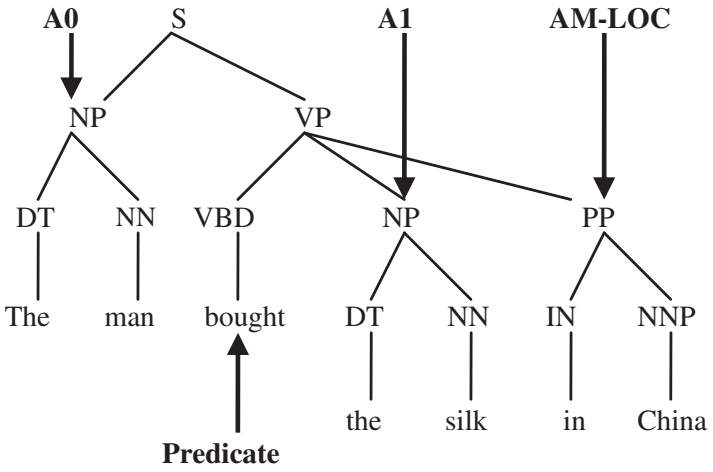

Fig. 1. A predicate annotation of semantic roles in a parse tree representation.

agent and a prototypical patient or theme, respectively. No consistent generalizations can be made across predicates for the remaining core arguments (i.e., A2–A5 and AA), though some effort has been made to consistently define roles across members of VerbNet (Kipper, Korhonen, Ryant, & Palmer, 2008) classes. Besides the core arguments, there are a set of 14 adjunct arguments (e.g., AM-MNR for manner, AM-LOC for locative and AM-TMP for temporal), which are universal to all predicates. In this paper, we will only consider PropBank due to its large scale, popularity and compatibility with the Penn TreeBank (Marcus, Marcinkiewicz, & Santorini, 1993). Fig. 1 shows an example of PropBank style annotation for the parse tree of the sentence "The man bought the silk in China" and the predicate "bought". In this example, a SRL system needs to label the sentence as "[$_{A0}$ The man] bought [$_{A1}$ the silk] [$_{AM\text{-}LOC}$ in China]", where given "bought" as the predicate in consideration, "The man" is labeled as the *agent* of the predicate, "the silk" is labeled as the *patient* of the predicate, and "in China" is labeled as the *locative* of the predicate.

Due to the competence of various semantic roles in capturing *who, what, whom, where, when, and how* information, SRL plays a critical role in NLP and has been widely applied in various NLP tasks, such as information extraction (Surdeanu, Harabagiu, Williams, & Aarseth, 2003), question answering (Narayanan & Harbabagiu, 2004), and co-reference resolution (Ponzetto & Strube, 2006). For example, Surdeanu et al. (2003) proposed an information extraction paradigm based on the predicate-argument structures in texts, which could be easily customized to new domains. Narayanan and Harbabagiu (2004) first used the SRL techniques to identify the predicate-argument structures and semantic frames from both the questions and documents, and then performed structured probability inference to extract answers for complex answers.

Generally, SRL takes a parse tree as input and determines proper labels for semantic arguments of a given predicate. Given a sentence and its parse tree, all predicates in the sentence can be easily recognized with simple heuristic rules or a classifier (Carreras & Màrquez, 2004, 2005), and for each recognized predicate, their semantic arguments are identified and labeled independently. From the learning algorithm viewpoint, SRL can be recast as a classification problem and consists of two sub-tasks: semantic argument identification, which identifies each syntactic constituent in a sentence as either a semantic argument of a given predicate or not, and semantic role classification, which classifies each identified semantic argument into a specific semantic role. With the availability of large annotated corpora such as FrameNet (Baker et al., 1998) and PropBank (Palmer et al., 2005), data-driven techniques, particularly machine-learning approaches, including both feature-based and tree kernel methods, have been extensively studied for SRL (Carreras & Màrquez, 2004, 2005; Gildea & Jurafsky, 2002; Moschitti, 2004; Moschitti, Pighin, & Basili, 2008). In feature-based methods, a flat feature vector is used to represent the parse tree structure[1] involving a particular predicate and its potential semantic argument. Typical features associated with the parse tree structure include phrase type, governing category, parse tree path, position, voice and head word. In tree kernel methods, a tree kernel function is used to directly measure the similarity between two parse tree structures without explicitly enumerating their substructures. Although feature-based methods have been consistently performing much better than tree kernel methods and represent the state-of-the-art in SRL, the problem with feature-based methods is that it is difficult for them to model syntactic structure information effectively. For example, the path feature widely used in feature-based methods is so sensitive to any change in a parse tree structure that a pair of parse tree structures will be represented by two different path features even if they differ only by one node (Zhang et al., 2007). In contrast, tree kernel methods have the potential to more effectively capture structured knowledge than feature-based methods. Moreover, it is well known that syntactic structured information in a parse tree plays an important role in feature-based SRL (Gildea & Jurafsky, 2002; Punyakanok, Roth, & Yih, 2005) and is thereby worth exploring in a more systematic way via tree kernel methods.

[1] In this paper, the term "parse tree structure" refers to the structure encapsulating the semantic relationship between a predicate and an argument (i.e., a syntactic constituent).

While it is difficult to further improve the performance of feature-based methods in SRL due to heuristic feature engineering, automatic feature engineering via the kernel methods provides a vital way to explore the relationship governing complex predicate-argument structures and gain useful insights into the underlying linguistic phenomena (Punyakanok et al., 2005). In the literature, various tree kernels have been proposed in NLP, such as the convolution tree kernel (Collins & Duffy, 2001), the PT kernel (Moschitti, 2006), the labeled order tree kernel (Kashima, 2007; Kashima & Koyanagi, 2002), the path-sensitive parse tree kernel (Zelenko, Aone, & Richardella, 2003), the path-sensitive dependency tree kernel (Culotta & Sorensen, 2004), the shortest path kernel over dependency trees (Bunescu & Mooney, 2005), and the LTAG-based tree kernel (Shen, Sarkar, & Joshi, 2003). As a representative tree kernel, the convolution tree kernel (CTK) implicitly takes sub-trees as its features and counts the number of common sub-trees as the similarity between two parse tree structures. This tree kernel has achieved the state-of-the-art in tree kernel-based SRL (Che, Zhang, Liu, & Li, 2006; Moschitti, 2004; Moschitti et al., 2008; Zhang et al., 2007) and other NLP applications, such as syntactic parsing (Collins & Duffy, 2001) and semantic relation extraction between named entities (Zhang, Zhang, Su, & Zhou, 2006; Zhou, Zhang, Ji, & Zhu, 2007). However, the standard CTK performs hard matching between two sub-trees without considering linguistic knowledge. This leads to two potential drawbacks: (1) The tree kernel fails to handle similar phrase structures (e.g., "buy a car" vs. "buy a red car"); (2) The tree kernel fails to differentiate various portions of parse trees (e.g., NP (noun phrase) in the subject position vs. NP in the object position).

This paper addresses the above-mentioned issues by proposing a tree kernel method for SRL with an enriched parse tree structure. This is done in two aspects. First, a context-sensitive convolution tree kernel with approximate matching mechanisms is presented to effectively capture the inherent structured knowledge in a parse tree. Second, an enriched parse tree structure is proposed to both well preserve the necessary structured information and effectively avoid noise by differentiating various portions of a parse tree structure when computing the tree kernel. Evaluation on the CoNLL'2005 shared task shows that both the new tree kernel and the enriched parse tree structure contribute much in SRL and our tree kernel method significantly outperforms the state-of-the-art tree kernel methods.

The layout of this paper is as follows. In Section 2, we review related work in more detail. Then, the enriched parse tree structure is introduced in Section 3 while the context-sensitive convolution tree kernel with approximate matching mechanisms is presented in Section 4. Section 5 shows the experimental results. Finally, we conclude our work in Section 6.

2. Related work

SRL has been drawing more and more attention in recent years and previous work in SRL can be classified into two categories: feature-based methods and tree kernel methods.

Feature-based methods have been widely employed in SRL and achieved much success. As a pioneer in feature-based SRL, Gildea and Jurafsky (2002) explored SRL on FrameNet with statistical and machine learning techniques, and proposed several widely used features, including phrase type, governing category, parse tree path, position, voice and head word. Since then, various informative features have been extensively studied with different learning techniques and tagging strategies, fueled by the availability of PropBank and the shared tasks of CoNLL-2004 and 2005. As a representative, Pradhan et al. (2005) made an extensive study on feature-based SRL through different feature sets, tagging strategies, and sentence inputs. They reported the performance of 84% in precision and 75% in recall by using automatic parse trees on PropBank. Nevertheless, the further improvement of performance has been limited by the difficulty of heuristic feature engineering and the bottleneck of syntactic parsing.

As an alternative to feature-based methods, kernel methods (Vapnik, 1998) have been proposed to implicitly explore features in a high dimensional space by employing a kernel to calculate the similarity between two objects directly. In particular, the kernel methods could be very effective at reducing the burden of heuristic feature engineering for structured objects in NLP research. This is because a kernel can measure the similarity between two discrete structured objects directly using the original representation of the objects instead of explicitly enumerating their features. Of special interest here, Moschitti (2004) pioneered the research in tree kernel-based SRL. He extracted the Predicate Argument Feature (PAF) portion from a parse tree, which includes salient substructures of the predicate-argument structure for SRL, and computed the similarity between two PAFs using the CTK. Moschitti et al. (2008) further explored various kinds of parse tree structures in tree kernel-based SRL and their complementary nature via composite kernels. Under the same framework, Che et al. (2006) further separated the PAF into a path portion and a Constituent Structure portion and combined two CTKs over these two separated portions using linear interpolation. It showed that such separation of the PAF into two different portions much increased the performance on the CoNLL'2005 shared task. Zhang et al. (2007) proposed a grammar-driven CTK with approximate matching to explore more linguistic knowledge in the kernel design for SRL. Evaluation on the CoNLL'2005 shared task verified its effectiveness in capturing approximate substructures by allowing grammar-driven approximate matching of substructures and nodes. In addition, Zhou et al. (2007) proposed a context-sensitive CTK, which enumerates both context-free and context-sensitive sub-trees by considering their ancestor node paths as their contexts. Evaluation on the ACE RDC 2003 and 2004 corpora showed that the context-sensitive tree kernel outperforms the standard CTK in the semantic relation extraction task.

Encouraged by the success of previous tree kernels, our tree kernel method further improves the state-of-the-art by (1) introducing an enriched parse tree structure via a differentiating strategy to better represent the structural information in a

parse tree; and (2) incorporating both context-sensitiveness and approximate matching into the popular CTK to better capture structural similarity between two parse tree structures. Evaluation on the CoNLL'2005 shared task shows that our tree kernel method significantly outperforms previous tree kernel methods and largely reduces the performance gap with the state-of-the-art feature-based methods in SRL.

3. Enriched parse tree structure

The semantic role relationship between a predicate and an argument is encapsulated by a parse tree. Thus, it is critical to understand which portion of a parse tree is important in tree kernel-based SRL. Representative parse tree structures in tree kernel-based SRL include PAF (Moschitti, 2004) and separated PAF (Che et al., 2006; Zhang et al., 2007).

Moschitti (2004) extracted the Predicate Argument Feature (PAF) portion from a parse tree as the parse tree structure, which includes the smallest substructure of the predicate, the argument and the shortest path between them. Fig. 2 illustrates the parse tree of the sentence "The man bought the silk in China" with the PAFs (the circled portions) as the parse tree structures of the predicate "bought" for semantic roles A0 and AM-LOC.

Che et al. (2006) and Zhang et al. (2007) further separated the PAF into a path portion, which contains the predicate and the shortest path, and a Constituent Structure portion, which contains the argument, and combined two CTKs over these two separated portions using linear interpolation. Fig. 3 illustrates the separated PAFs as the parse tree structures for semantic roles A0 and AM-LOC. Evaluation on the CoNLL'2005 shared task shows that such separating strategy much increases the performance by avoiding the wrong matching between different portions of two PAFs.

In this paper, we propose a simple but efficient differentiating strategy to avoid the wrong matching across different portions of two parse tree structures by explicitly differentiating various portions in a parse tree structure using different labels instead of separating them into several portions. In particular, we differentiate the PAF into four portions: the predicate, the argument, the head of the argument and the shortest path. Here, the head of an argument is differentiated from the remaining part of the argument since it includes critical information about the argument. Fig. 4 illustrates the differentiated PAFs as the parse tree structures for semantic roles A0 and AM-LOC, where the additive labels "1", "2", "3" and "4" are used to indicate the predicate, the argument, the argument head and the shortest path portions, respectively. For example, "NP-24" in the A0 annotation means that the "NP" node occurs in both the argument portion and the shortest path portion.

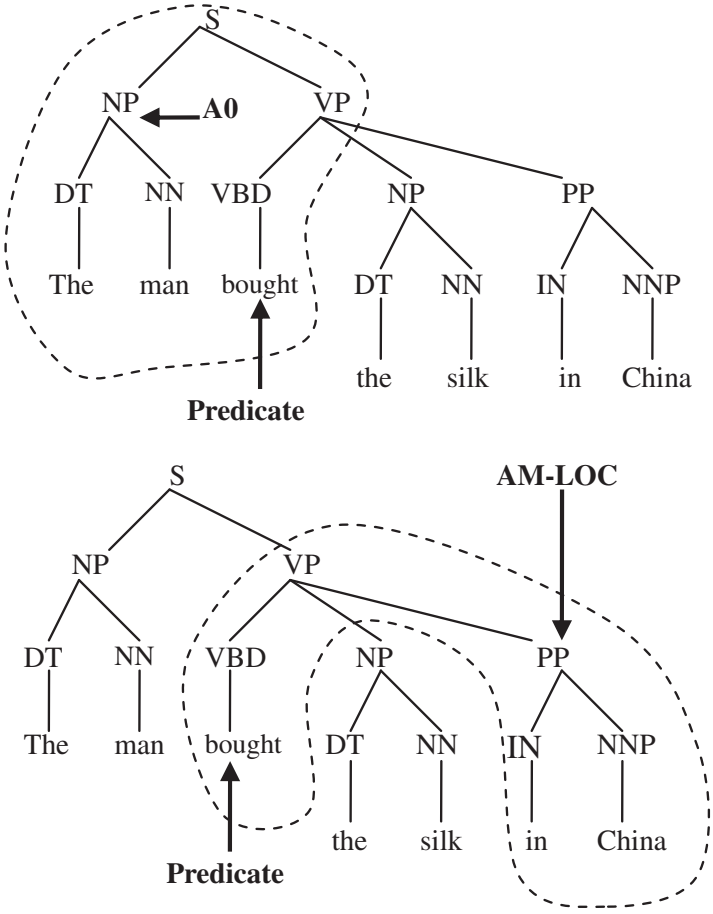

Fig. 2. PAFs (the circled portions) for semantic roles A0 and AM-LOC of the predicate "bought" in the sentence "The man bought the silk in China".

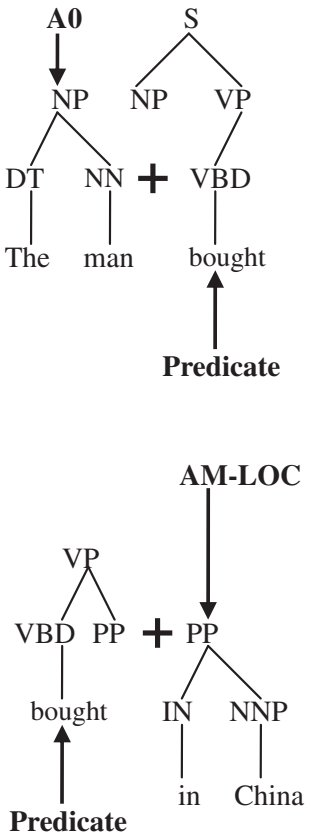

Fig. 3. Separated PAFs for semantic roles A0 and AM-LOC of the predicate "bought" in the sentence "The man bought the silk in China".

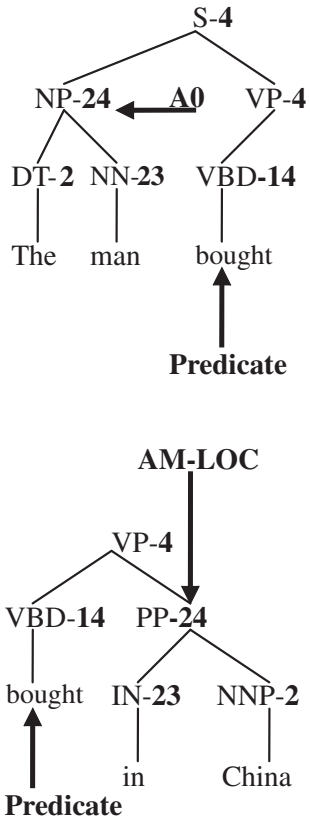

Fig. 4. Differentiated PAFs for semantic roles A0 and AM-LOC of the predicate "bought" in the sentence "The man bought the silk in China".

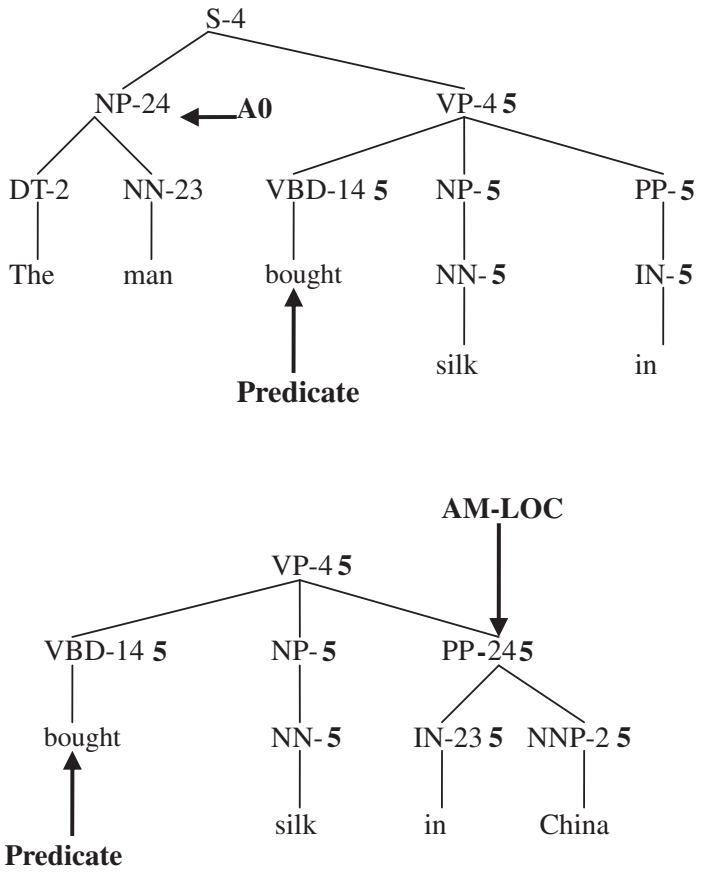

Fig. 5. Differentiated PAF + SubCats for semantic roles A0 and AM-LOC of the predicate "bought" in the sentence "The man bought the silk in China".

Although the sub-categorization structure of the predicate is proven very useful in feature-based SRL, it is interesting to note that such structure has not been well explored in the state-of-the-art tree kernel-based SRL. This may be due to the fact that simple inclusion of such predicate sub-categorization structure may introduce too much noise in the tree kernel computation and thus harm the performance. We hope that the differentiation of various portions of a parse tree structure can effectively avoid this problem and such predicate sub-categorization structure can be effectively captured via the differentiating strategy. Fig. 5 illustrates the differentiated PAF + SubCat as the parse tree structures for semantic roles A0 and AM-LOC. Please note that the head word of each child in the predicate sub-categorization structure and its POS (part-of-speech) are also included, with the label "5" indicating the predicate sub-categorization portion.

4. Context-sensitive convolution tree kernel with approximate matching

Given two parse trees (or two parse tree structures), we now study how to measure the similarity between them, using a convolution kernel. A convolution kernel (Haussler, 1999) aims to capture structured knowledge in terms of substructures. As a specialized convolution kernel, the convolution tree kernel (Collins & Duffy, 2001) $K_{CTK}(T_1, T_2)$ ('CTK' for convolution tree kernel) counts the number of common sub-trees (substructures) as the syntactic structure similarity between two parse trees T_1 and T_2:

$$K_{CTK}(T_1, T_2) = \sum_{n_1 \in N_1, n_2 \in N_2} \Delta(n_1, n_2) \tag{1}$$

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 nodes[2] and is computed recursively as follows:

(1) If the context-free productions (i.e., context-free grammar rules) at n_1 and n_2 do not match exactly, then $\Delta(n_1, n_2) = 0$; otherwise go to 2.
(2) If both n_1 and n_2 are POS tags, then $\Delta(n_1, n_2) = \lambda$; otherwise go to 3.
(3) Calculate $\Delta(n_1, n_2)$ recursively as:

[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.

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

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

Although the above standard CTK has been successfully employed in many NLP applications, it has two drawbacks: (1) the sub-trees involved in the tree kernel computation are context-free (i.e., they do not consider the information outside the sub-trees); and (2) it only allows exact matching of sub-trees.

In order to resolve the above two shortcomings, this paper proposes a new CTK by equipping the standard CTK with both context-sensitiveness and approximate matching.

4.1. Context-sensitiveness

To equip the CTK with context-sensitiveness, our new tree kernel takes ancestral information (i.e., the root node path) of sub-trees in the given two parse trees $T[1]$ and $T[2]$ into consideration by modifying Eq. (1) as follows:

$$K_{CTK}(T[1], T[2]) = \sum_{i=1}^{m} w_i \cdot \sum_{\substack{n_1^i[1] \in N_1^i[1] \\ n_1^i[2] \in N_1^i[2]}} \Delta(n_1^i[1], n_1^i[2]) \quad (1')$$

where $n_1^i[j] = (n_1 n_2 \ldots n_i)[j]$ is a root node path[3] (from current node up to the root node of the tree) with length i in tree $T[j]$ taking into account the $i-1$ ancestral nodes $n_2^i[j]$ of $n_1[j]$ in $T[j]$. $\Delta(n_1^i[1], n_1^i[2])$ measures the common context-sensitive sub-trees with root node paths $n_1^i[1]$ and $n_1^i[2]$, m defines the maximal length of a root node path and w_i is the weight for a context-sensitive sub-tree with a root node path of length i.

Obviously, the standard CTK is a special case of our tree kernel (with $m = 1$ and $w_1 = 1$ in Eq. (1′)). In particular, 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.

Concerning SRL, the advantage of context-sensitiveness is obvious. For example, NPs (noun phrases) occupy 53% of arguments in Section 23 of PropBank. Among them, 95% of NPs have the semantic role of A0 when its parent node is SINV (inverted declarative sentence), 97% of NPs have the semantic role of either A0 or A1 when its parent node is S (simple declarative clause), and 77% of NPs have the semantic role of A1 when its parent node is VP (verb phrase). That is, an NP can be labeled more accurately with the knowledge of its parent node. This phenomenon also applies to other types of phrases.

4.2. Approximate matching

The standard CTK only allows exact matching. To further introduce approximate matching, our tree kernel changes the way of computing $\Delta(n_1^i[1], n_1^i[2])$ to allow insertion/deletion/substitution of tree nodes in the substructures:

(1) If the context-sensitive productions (i.e., context-sensitive grammar rules) at $n_1^i[1]$ and $n_1^i[2]$ do not match (either exactly or approximately), then $\Delta(n_1^i[1], n_1^i[2]) = 0$; otherwise go to 2.
(2) If both $n_1[1]$ and $n_1[2]$ are POS tags, then $\Delta(n_1^i[1], n_1^i[2]) = \lambda$; otherwise go to 3.
(3) Calculate $\Delta(n_1^i[1], n_1^i[2])$ recursively as:

$$\Delta(n_1^i[1], n_1^i[2]) = \lambda \cdot \lambda_1^{\#InsDels} \cdot \lambda_2^{\#Subs} \cdot \prod_{k=1}^{\#ch(x_1^i[1])} (1 + \Delta(ch(x_1^i[1], k), ch(x_1^i[2], k))) \quad (2')$$

where #InsDels is the number of insertions/deletions of optional tree nodes and #Subs is the number of substitutions among similar tree nodes while $\lambda_1 (1 < \lambda_1 < 1)$ and $\lambda_2 (1 < \lambda_2 < 1)$ are their respective weights. $x_1^i[j]$ is the best matched child sequence between $n_1^i[1]$ and $n_1^i[2]$ of length $\#ch(x_1^i[j])$ with approximate matching mechanisms. Here, $ch(x_1^i[j], k])$ is the kth child of $x_1^i[j]$.

Obviously, the number of insertions/deletions/substitutions can be easily determined by computing the least edit distance (weighted) between two productions using dynamic programming. In particular, all the children of a node are optional tree nodes except its head child since head children normally contain critical information and should be same or similar for two productions to be matched (either exactly or approximately). This means insertions and deletions only apply to non-head children while substitutions only happen in the same group of similarly-functioned tree nodes, e.g., the POSs for singular nouns and plural nouns. Moreover, in the differentiated parse tree structures, the extended labels should be matched exactly. For example, NN-24 can match approximately with NNS-24 but cannot match with NN-2, in order to avoid wrong matching across different portions of the differentiated parse tree structures.

[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.

Concerning SRL, the standard CTK requires exact matching between two contiguous phrase structures. This constraint may be too strict and makes the standard CTK fail to handle similar phrase structures (e.g., "a car" vs. "a *red* car") and quasi-synonymic grammar tags (e.g., the POS variations between "*high*/JJ degree/NN" and "*higher*/JJR degree/NN"). In addition, some rewritten rules are generalizations of others. For example, "NP → DT JJ NN" is a specialized version of "NP → DT NN". The same applies to POSs and other grammatical tags. The approximate matching mechanism provides an effective way to capture such inherent similarities.

4.3. Comparison

To provide better understanding of our tree kernel, this section compares it in detail with related tree kernels, such as the standard convolution tree kernel (Collins & Duffy, 2001), the partial tree (PT) kernel (Moschitti, 2006), the labeled order tree kernel (Kashima & Koyanagi, 2002; Kashima, 2007), the LTAG-based tree kernel (Shen et al., 2003), the path-sensitive parse tree kernel (Zelenko et al., 2003), the path-sensitive dependency tree kernel (Culotta & Sorensen, 2004), the shortest path kernel over dependency trees (Bunescu & Mooney, 2005), the context-sensitive convolution tree kernel (Zhou et al., 2007) and the grammar-driven convolution tree kernel (Zhang et al., 2007).

The three conventional convolution tree kernels, i.e., the standard (Collins & Duffy, 2001), the context-sensitive (Zhou et al., 2007) and the grammar-driven (Zhang et al., 2007) ones, are all particular cases of our tree kernel. First, our tree kernel considers not only exact matching but also approximate matching. The main difference on approximate matching between our tree kernel and the grammar-driven convolution tree kernel is that approximate matching in our tree kernel is measured by the edit distance via dynamic programming while approximate matching in the grammar-driven convolution tree kernel is derived from the partial matching algorithm proposed in the partial tree (PT) kernel (Moschitti, 2006) by imposing certain grammar-driven constraints. Moreover, our tree kernel is context-sensitive in that it takes ancestral information into consideration when comparing two sub-trees. The main difference on context-sensitiveness between our tree kernel and the context-sensitive convolution tree kernel is that the context-sensitive convolution tree kernel treats sub-trees with different ancestor information equally while our tree kernel treats them differently and benefits much from this. Finally, an enriched parse tree structure is proposed in this paper to both well preserve the necessary structured information and effectively avoid noise by differentiating various portions of a parse tree structure when computing the tree kernel.

Compared with the partial tree (PT) kernel (Moschitti, 2006) which allows partial matching between sub-trees, our tree kernel generates much fewer substructures. The main difference is that our tree kernel constrains the similarity of two productions by requiring exact matching or similar substitution between their head children while the PT kernel does not have this constraint and hence allows many non-linguistically motivated substructures to be matched. This may compromise the performance in SRL (Moschitti, 2006) since some of the generated non-grammatical substructures may be noisy due to the lack of linguistic interpretations and constraints. Another difference is that the PT kernel does not allow approximate node matching. As a result, our tree kernel better exploits linguistic knowledge than the PT kernel.

The labeled order tree kernel (Kashima & Koyanagi, 2002; Kashima, 2007) generalizes the convolution kernel over labeled order trees. It is much more adaptable than the PT kernel and can explore much larger substructures than the PT kernel through exploiting label mutations and elastic structure matching. However, the same as the PT kernel, the labeled order tree kernel faces the same issues (over-generation of non-grammatical substructures) as the PT kernel when used in NLP applications.

Both the path-sensitive parse tree (Zelenko et al., 2003) and dependency tree (Culotta & Sorensen, 2004) kernels match nodes from roots to leaf nodes recursively layer by layer in a top-down manner with the former working on parse trees and the latter on dependency trees. However, these two kernels impose a strong constraint on tree kernel computation: matchable nodes should have an identical path of ascending nodes from the roots to the current nodes. For the shortest path kernel over dependency trees (Bunescu & Mooney, 2005), it simply counts the number of common word classes at each position in the shortest paths between two constituents in dependency trees. It is worth noting that such shortest paths have been widely and successfully used as features in the state-of-the-art feature-based SRL. Compared with the above three kernels, convolution tree kernels are more flexible and robust. Evaluation on the ACE RDC corpus (ACE 2002–2005) of the semantic relation extraction task (Zhang et al., 2006) shows that the standard CTK significantly outperforms the above three kernels.

The LTAG-based tree kernel (Shen et al., 2003) was proposed to utilize LTAG-based features in parse tree re-ranking. It works on LTAG derivation trees by first deriving a LTAG derivation tree from a parse tree and then computing the similarity between two LTAG derivation trees instead of two parse trees.

5. Experimentation

We have systematically evaluated our tree kernel methods in SRL on the CoNLL'2005 shared task.

5.1. Experimental setting

The CoNLL'2005 shared task data (Carreras & Màrquez, 2005) consists of the Wall Street Journal part of the Penn TreeBank (Marcus et al., 1993), with predicate annotations of semantic roles from the PropBank I corpus (Palmer et al., 2005). There are

35 roles including seven core (A0–A5, AA), 14 adjunct (AM-) and 14 reference (R-) arguments, which occupy about 70%, 27% and 3% of arguments on the CoNLL'2005 shared task data, respectively. Among the seven core arguments, A0, A1 and A2 occupy more than 95% of arguments. Among the 14 adjunct arguments, AM-ADV, AM-DIS, AM-LOC, AM-MNR, AM-MOD, AM-NEG and AM-TMP occupy more than 85% of arguments. Among the 14 reference arguments, R-A0, R-A1 and R-AM-TMP occupy nearly 90% of arguments. However, due to the minority of the 14 reference arguments, their impact on the whole performance can be neglected and we will not explore them in further detail. Please refer to Appendix A for a brief overview of semantic roles on the CoNLL'2005 shared task. As defined by the shared task, we use Sections 02–21 for training, Section 24 for development and Section 23 for testing, respectively. In particular, we only use syntactic constituents from a single automatic parse tree as the labeling units to form the labeled arguments. Here, all the evaluations are done on the automatic parse trees delivered by the Charniak parser (Charniak, 2000), adopting the common four-stage approach:

- *Pruning*: An effective pruning algorithm is applied to filter out the constituents that are not semantic arguments to the predicate, using four simple linguistics-driven heuristic rules: (a) if a constituent overlaps with the given predicate, it is filtered; (b) non-POS constituents with only one child are filtered; (c) if a constituent is kept, its head child is filtered; and (d) all the children of a base phrase constituent (i.e., all of its children are POS nodes) are filtered. Statistics on the training data shows that our pruning algorithm can effectively filter out ~75% of non-arguments at the cost of wrongly filtering out 1.5% semantic arguments.
- *Semantic argument identification*: The remaining candidates are identified as either arguments or not using a binary classifier.
- *Semantic role classification*: The identified arguments are classified into one of the semantic roles using a multi-category classifier.
- *Post processing*: Similar to Pradhan et al. (2005), overlapped semantic roles are validated by only keeping non-overlapped ones with largest margins computed in Step 3.

In this paper, the SVM-Light Toolkit (Joachims, 1998) is selected as our classifier in both the semantic argument identification stage and the semantic role classification stage. Since SVM-Light is a binary classifier, we adopt the *one vs. others* strategy to handle the multi-category classification problem in the semantic role classification stage and select the label with the largest margin as the final output. To speed up the training process, all the free parameters are fine-tuned on the development data (Section 24) using Section 02–05 as the training data.

For approximate matching, we only consider four groups of similarly-functioned tree nodes: (1) adjective group: ADJP, JJ, JJR, JJS; (2) adverb group: ADVP, RB, RBR, RBS; (3) noun group: NP, NN, NNS, NNP, NNPS, NAC, NX; and (4) verb group: VB, VBD, VBG, VBP, VBZ, VBN and VP (in active voice only). Since voice information is very indicative in differentiating A0 and A1, VBNs and VPs in the passive voice are determined and differentiated from other verbs tags. Please refer to Appendix B for a brief overview of POS and phrase labels employed in approximate matching.

Finally, all the performances are measured on both the overall semantic role labeling task using F-measure and the semantic role classification task using Accuracy (over the matched semantic arguments) on a single automatic parse tree only returned by the Charniak parser (Charniak, 2000). To see whether an improvement is significant, we also conduct significance testing using the paired *t*-test. In this paper, '>>>', '>>', and '>' denote *p*-values of the risk that the improvement is generated by random less than 0.01, 0.01–0.05, and greater than 0.05, which mean significantly better, moderately better, and slightly better, respectively.

5.2. Experimental results and discussion

The experiments are done in the following order:

(1) Evaluating various kinds of parse tree structures over the standard CTK to compare the impact of parse tree structures in SRL.
(2) Evaluating various CTKs over a parse tree structure to show the impact of context-sensitiveness and approximate matching.
(3) Comparing our tree kernel method with the state-of-the-art tree kernel methods to show the progress achieved by our tree kernel method.
(4) Integrating our tree kernel method with a state-of-the-art feature-based method via linear interpolation to show their complementary nature.

5.2.1. Various parse tree structures over the standard CTK

Table 1 compares various parse tree structures, using the standard CTK with the SVM-Light regularization parameter C and the CTK decay factor λ fine-tuned to 2.4 and 0.4 (hereafter in this paper) respectively. It shows that

(1) Similar to Che et al. (2006), separation of the PAF into a PATH portion and a Constituent Structure portion (PAF vs. separated PAF) much improves the performance by 1.63% (>>>) and 1.76% (>>>) in F-measure and Accuracy for

Table 1
Comparison of different parse tree structures using the standard CTK on a single automatic parse tree.

Parse tree structure	F-measure (Accuracy %)
PAF	70.73 (84.31)
Separated PAF	72.36 (86.07)
Differentiated PAF	73.51 (87.28)
PAF + SubCat	70.18 (83.62)
Separated PAF + SubCat	72.66 (86.39)
Differentiated PAF + SubCat	74.25 (88.02)

Note: (1) the figures outside the parentheses indicate the F-measure for the overall semantic role labeling task while the figures inside the parentheses are the Accuracy for the semantic role classification task only (this also applies to Tables 2–4); and (2) Differentiated PAFs/PAF + SubCats, as shown in Figs. 4 and 5, will become PAFs/PAF + Subs by removing the additive labels on the tree nodes.

semantic role labeling and classification respectively. Here, the kernel function between two separated PAFs is computed by linearly interpolating the kernel functions on the two separated portions:

$$K_{CTK\text{-}SPAF} = \gamma \cdot K_{CTK\text{-}CS} + (1-\gamma) \cdot K_{CTK\text{-}PATH}$$

where $K_{CTK\text{-}CS}$ and $K_{CTK\text{-}PATH}$ measure the kernel functions on the Constituent Structure portion and the PATH portion respectively, using the standard CTK. Here, γ $(0 \leqslant \gamma \leqslant 1)$ is fine-tuned to 0.5.

(2) Differentiating various portions of PAFs much improves the performance by 1.15% (>>>) and 1.21% (>>>) in F-measure and Accuracy for semantic role labeling and classification respectively, over separating different portions of PAFs (differentiated PAF vs. separated PAF). In total, differentiated PAFs significantly outperform PAFs by 2.78% (>>>) and 2.97% (>>>) in F-measure and Accuracy for semantic role labeling and classification respectively.
(3) Simple inclusion of the predicate sub-categorization structure into the widely used PAF decreases the performance by 0.55% (>>) and 0.69% (>>) in F-measure and Accuracy for semantic role labeling and classification respectively (PAF vs. PAF + SubCat).
(4) Separated inclusion of the predicate sub-categorization structure into the separated PAFs slightly improves the performance by 0.30% (>) and 0.32% (>) in F-measure and Accuracy for semantic role labeling and classification respectively (separated PAF vs. separated PAF + SubCat). Similar to separated PAFs, the kernel function between two separated PAF + SubCats is computed by linearly interpolating the kernel functions on the three separated portions:

$$K_{CTK\text{-}SPAF+SubCat} = \alpha \cdot K_{CTK\text{-}CS} + \beta \cdot K_{CTK\text{-}PATH} + \gamma \cdot K_{CTK\text{-}SubCat}$$

where $K_{CTK\text{-}CS}$, $K_{CTK\text{-}PATH}$ and $K_{CTK\text{-}SubCat}$ measure the kernel functions on the constituent structure, the PATH and the predicate sub-categorization portions respectively, using the standard CTK. Moreover, α, β and γ $(0 \leqslant \alpha, \beta, \gamma \leqslant 1)$ are fine-tuned to 0.4, 0.4 and 0.2 respectively.

(5) Differentiated inclusion of the predicate sub-categorization structure into the differentiated PAFs improves the performance by 0.74% (>>) and 0.74% (>>) in F-measure and Accuracy for semantic role labeling and classification respectively (differentiated PAF vs. differentiated PAF + SubCat).
(6) Differentiating various portions of PAF + SubCat (differentiated PAF+SubCat vs. separated PAF + SubCat) much improves the performance by 1.59% (>>>) and 1.63% (>>>) in F-measure and Accuracy for semantic role labeling and classification respectively, over separating different portions of PAF + SubCat.

The above narrative suggests that

(1) Simple inclusion of important structural information, such as the predicate sub-categorization structure, into the widely used PAF structure may introduce too much noise and thus decreases the performance, though the predicate sub-categorization structure has been proven very useful in feature-based SRL.
(2) Separation of a PAF into different portions and further inclusion of important structural information, such as the predicate sub-categorization structure, are useful, due to that the separation strategy can much avoid wrong matching between different portions of two parse tree structures by only allowing matching inside the same portion between two parse tree structures.
(3) The differentiating strategy can better model the substructure similarity of two parse tree structures than the separating strategy. This may be due to that the separating strategy fails to capture the substructure similarity across different portions while the differentiating strategy cannot only avoid wrong matching between different portions but also capture the substructure similarity across different portions. Although minor diathesis alternations may lead

to different tree representations, they normally do not change the belonging of a node to a particular portion in the differentiating strategy and thus its enhanced label. Another advantage of the differentiating strategy over the separating strategy is that the differentiating strategy has less free parameters and is thus much easier to fine-tune.

Further examination of the experimental results shows that both core and adjunct arguments benefit from differentiated PAF (differentiated PAF vs. separated PAF). In particular, the three major core arguments (A0, A1 and A2) contribute nearly 80% to the performance improvement, mostly due to the differentiation of the argument head from the argument. Such differentiation emphasizes the argument head and thus the semantic relationship between the given predicate and the argument can be better captured. In addition, we find that inclusion of sub-categorization (differentiated PAF + SubCat vs. differentiated PAF) benefits core arguments most while its impact on adjunct arguments can be neglected. This is mainly due to the following reasons: (1) Sub-categorization usually implies such information like whether the predicate has an object or not. This is crucial for a core argument outside the VP (verb phrase) headed by the predicate. Taking the two sentences "He opens the door" and "The door opens" as examples, it is easy to label "He" as A0 (i.e., the agent of predicate "open") and to label "The door" as A1 (i.e., the patient of predicate "open") if the sub-categorization information of predicate "open" is available. (2) Adjunct arguments are usually self-explaining and independent from such sub-categorization information, such as whether the predicate has an object or not.

5.2.2. Various CTKs over a parse tree structure

Table 2 compares various CTKs using the differentiated PAF + SubCat as the parse tree structure. It shows that

(1) The context-sensitiveness alone (CTK vs. CTK with context-sensitiveness) increases the performance by 1.13% (>>) and 1.26% (>>) in F-measure and Accuracy for semantic role labeling and classification respectively. Here, m is fine-tuned to 3 while w_1, w_2 and w_3 are fine-tuned to 0.7, 0.2 and 0.1 respectively.
(2) The approximate matching alone (CTK with approximate matching) increases the performance by 1.32% (>>>) and 1.51% (>>>) in F-measure and Accuracy for semantic role labeling and classification respectively.
(3) Integration of both context-sensitiveness and approximate matching into the standard CTK increases the performance by 1.88% (>>>) and 2.23% (>>>) in F-measure and Accuracy for semantic role labeling and classification respectively.

This suggests that consideration of context-sensitive sub-trees is useful in tree kernel-based SRL and that approximate matching in terms of linguistic intuition is very useful due to both exact and approximate matching. Moreover, it is interesting to note that the enriched parse tree structure over the widely used PAF contributes much more than the new context-sensitive CTK with approximate matching over the standard CTK. This indicates that SRL can benefit much from a proper parse tree structure in tree kernel methods.

As discussed in Sections 4.1 and 4.2, context-sensitiveness extends the substructure space enumerated in the tree kernel computation and approximate matching provide a mechanism to better model the similarity between two similar substructures. Further examination of the experimental results also justifies earlier reasoning. It shows that the two major core arguments A0 and A1 benefit much from context-sensitiveness and approximate matching and contribute about 70% of the performance improvement. This is mostly due to the introduction of the ancestral information (via context-sensitiveness) in modeling the semantic role dependence of an argument (in particular in the form of NP) on its parental node and approximate matching in better measuring similar substructures. It also shows that the performance improvements on those major adjunct arguments vary. While AM-ADV, AM-MNR and AM-TMP contribute most, the performances of AM-MOD, AM-NEG and AM-DIS are quite stable for different tree kernels, due to the facts that: (1) Adjunct arguments of AM-MOD and AM-NEG are usually short and appear next to the predicate. Thus, they are easy to detect with the performance already up to 95–96 in F-measure using the standard CTK. As a result, it is difficult to further improve their performance. (2) Adjunct arguments of AM-DIS usually map to discourse makers, such as "however", "in addition" and "for example". They are quite insensitive to the variations in a parse tree structure.

5.2.3. Comparison of different tree kernel methods

Table 3 compares our tree kernel with previous best-reported tree kernels (Che et al., 2006; Moschitti, 2004; Zhang et al., 2007) in SRL. Benefiting from the fact that our tree kernel method integrates the advantages of several state-of-the-art kernel

Table 2
Comparison of different CTKs using the differentiated PAF + SubCat as the parse tree structures on a single automatic parse tree.

Tree kernel	F-measure (Accuracy %)
CTK	74.25 (88.02)
CTK with context-sensitiveness	75.38 (89.28)
CTK with approx. matching	75.57 (89.63)
CTK with context-sensitiveness and approx. matching	76.13 (90.25)

Table 3
Comparison of the state-of-the-art kernel methods in SRL on a single automatic parse tree.

Kernel method	F-measure (Accuracy %)
Moschitti (2004)	70.73 (84.33)
Che et al. (2006)	72.36 (86.07)
Zhang et al. (2007)	74.17 (88.04)
Ours	76.13 (90.25)

Note: (1) Moschitti (2006) applied CTK on PAFs to compute the kernel function; (2) Che et al. (2006) combined two CTKs on separated PAFs via linear interpolation; (3) Zhang et al. (2007) combined two CTKs with approximate matching on separated PAFs via linear interpolation; (4) Ours applied the context-sensitive CTK with approximate matching on differentiated PAF + SubCats to compute the kernel function.

Table 4
Complementary nature and comparison of our kernel method with the state-of the-art feature-based methods on a single automatic parse tree.

Methods	F-measure (Accuracy %)
Ours: composite kernel	79.64 (93.25)
Ours: tree kernel	76.13 (90.25)
Ours: feature-based	77.56 (91.85)
Surdeanu et al. (2005): feature-based	76.46 (–.–)
Zhang et al. (2007): feature-based	77.00 (89.92)

Note: (1) This comparison is done using a single automatic parse tree only; (2) Surdeanu et al. (2005) is the best individual system using a single automatic parse tree. Although some systems on the CoNLL'2005 shared task achieve better performance than Surdeanu et al. (2005), they use some kinds of combination so that they are not included here in comparison.

methods, including the above three, we are able to fairly compare them using the similar experimental settings as much as possible by reconfiguring our proposed tree kernel. Not surprisingly, it shows that our tree kernel significantly (>>>) outperforms all the previous best-reported tree kernels.

5.2.4. Linear interpolation of a tree kernel method and a feature-based method

In the literature, feature-based methods have been performing consistently much better than tree kernel methods on SRL. One may ask: can tree kernel methods close the performance gap and catch up with feature-based methods? Or can feature-based methods swamp the gain that tree kernel methods add? To better investigate the complementary nature between feature-based methods and tree kernel methods, we have implemented a state-of-the-art feature-based method using the linear kernel, which achieves the state-of-the-art F-measure of 77.56 with the same experimental setting,[4] and linearly interpolated with the above context-sensitive CTK with approximate matching using the differentiated PAF + SubCat as the parse tree structure:

$$K_{Comp} = \gamma \cdot K_{Linear} + (1 - \gamma) \cdot K_{CTK}$$

where K_{Linear} is the linear kernel (normalized) and K_{CTK} is our proposed CTK. The weight γ $(0 \leqslant \gamma \leqslant 1)$ is fine-tuned to 0.6. Table 4 shows that these two kinds of kernels are quite complementary and such linear interpolation achieves the F-measure and Accuracy of 79.64% and 93.25% on the overall semantic role labeling task and the semantic role classification task, respectively. This means that feature-based methods cannot swamp the gain that tree kernel methods add and SRL can benefit much from the research on tree kernel methods. Table 4 also shows that our tree kernel method achieves close performance with the state-of-the-art feature-based methods. Although our tree kernel method still lags a bit behind the state-of-the-art feature-based methods due to the latter's ability of covering various knowledge sources which may be hard for tree kernel methods, our exploration of tree kernel methods in SRL is much promising in their ability of better capturing structural parse tree information.

[4] This is achieved through extensive feature engineering by incorporating the widely used unigram features, including various parse tree features used in the literature (Gildea & Jurafsky, 2002; Pradhan, Ward, Hacioglu, Martin, & Jurafsky, 2004; Pradhan et al., 2005; Toutanova et al., 2005; Punyakanok et al., 2005), and the bigram and trigram features as described in Liu, Che, and Li (2007). In addition, only the feature-based linear kernel is applied in this paper since the polynomial kernel fails to improve the performance. This may be due to the extensive inclusion of bigram and trigram features.

6. Conclusion

It is well known that structured information in a parse tree plays a critical role in many advanced NLP applications (including SRL) and that kernel methods have the potential to more effectively capture structured knowledge than feature-based methods. However, there is still a big performance gap in SRL between kernel methods and feature-based methods. In order to reduce this gap, this paper has attempted to advance kernel-based SRL by (1) proposing a new tree kernel via incorporating context-sensitiveness and approximate matching into the standard CTK; and (2) proposing an enriched parse tree structure via a differentiating strategy.

Evaluation on the CoNLL'2005 shared task shows that our tree kernel method significantly outperforms previous best-reported tree kernel methods and largely reduces the performance gap between tree kernel methods and feature-based methods. To the best of our knowledge, this is the first research of tree kernel methods in SRL, which achieves such close performance with feature-based methods. Although there is still some performance gap between our tree kernel method and the state-of-the-art feature-based methods, these two kinds of methods are much complementary in that the tree kernel method can much better explore structural parse tree information. It also shows that introducing context-sensitiveness and approximate matching into the standard CTK much improve the performance while the enriched parse tree structure via the differentiating strategy not only avoids wrong matching between different portions of parse tree structures but also better incorporates structured information. Finally, it is interesting to note that the enriched parse tree structure over the widely used PAF contributes much more than the new context-sensitive convolution tree kernel with approximate matching over the standard convolution tree kernel.

An alternative to tree kernel-based SRL on the constituent parse tree structure is the one on the dependency tree structure. In fact, dependency trees are in their nature much closer to predicate-argument structures. However, it is still a big challenge on how to effectively employ tree kernels in SRL from the dependency tree perspective. This may be due to the fact that the dependency tree structure is too fine-grained. That is, all the nodes in the dependency tree structure are words. This makes the dependency tree kernels suffer from the severe data sparseness problem. For example, while both the CoNLL'2008 and CoNLL'2009 shared tasks aim at SRL from the dependency tree perspective (Hajic, Ciaramita, Johansson, et al., 2009; Surdeanu, Johansson, Meyers, Màrquez, & Nivre, 2008), all the 5-best reported systems on both shared tasks employ feature-based methods instead of dependency tree kernel methods. In contrast, the constituent parse tree structure employed in this paper benefits much from its layered abstraction of introducing non-terminal nodes, i.e., POSs (such as NN for noun and VB for verb) and phrase labels (such as NP for noun phrase and VP for verb phrase), and thus is more flexible. This has also been verified by previous work on a similar task, semantic relation extraction between named entities, instead of SRL between a given predicate and an argument.

For the future work, we will explore more effective ways of representing the predicate-argument structures in SRL and new kernels in better modeling such predicate-argument structures. Moreover, we will apply our new kernel method in other advanced NLP applications, such as semantic relation extraction and co-reference resolution.

Acknowledgements

This research is supported by Projects 60873043, 60873150, 60970056 and 90920004 under the National Natural Science Foundation of China.

Appendix A. Semantic roles involved in the CoNLL'2005 shared task

Following is a brief overview of semantic roles involved in the CoNLL'2005 shared task. For details, please refer to Palmer et al. (2005).

(1) 7 Core arguments (A0–A5, AA): Normally, A0 and A1 denotes the agent and the patient, respectively. A2–A5, and AA are predicate-specific. For each verb, PropBank defines a set of possible roles (called roleset).
(2) 14 Adjunct arguments (AM-): Generally, these arguments may be taken by any verb and are classified into AM-ADV for general-purpose, AM-CAU for cause, AM-DIR for direction, AM-DIS for discourse maker, AM-EXT for extent, AM-LOC for locative, AM-MNR for manner, AM-MOD for modal verb, AM-NEG for negation maker, AM-PNC for purpose, AM-PRD for predication, AM-REC for reciprocal and AM-TMP: temporal.
(3) 14 Reference arguments (R-): Representing those arguments realized in other parts of the sentence. The role of a reference is the same as the role of the referenced argument. The label is an R- tag prefixed to the label of referent, e.g., R-A0, R-A1 and R-AM-TMP.

Appendix B. Part-of-speech and phrase tags employed in approximate matching

Following is a brief overview of part-of-speech and phrase tags employed in approximate matching. For details, please refer to Marcus et al. (1993).

(1) Adjective group: ADJP for adjective phrase, JJ for adjective, JJR for adjective (comparative), JJS for adjective (superlative).
(2) Adverb group: ADVP for adverb phrase, RB for adverb, RBR for adverb (comparative) and RBS for adverb (superlative).
(3) Noun group: NP for noun phrase, NN for noun (singular or mass), NNS for noun (plural), NNP for proper noun (singular), NNPS for proper noun (plural), NX used within certain complex NPs to mark the head of the NP and NAC for not a constituent.
(4) Verb group: VP for verb phrase, VB for verb (base form), VBD for verb (past tense), VBG for verb (gerund or present participle), VBN for verb (past participle), VBP for verb (non-3rd person singular present) and VBZ for verb (3rd person singular present).

References

Baker, C. F., Fillmore, C. J., & Lowe, J. B. (1998). The Berkeley FrameNet Project. *COLING-ACL'1998*.
Bunescu, R., & Mooney, R. J. (2005). A shortest path dependency kernel for relation extraction. *HLT/EMNLP'2005*.
Carreras, X., & Màrquez, L. (2004). Introduction to the CoNLL-2004 shared task: Semantic role labeling. *CoNLL'2004*.
Carreras, X., & Màrquez, L. (2005). Introduction to the CoNLL-2005 shared task: Semantic role labeling. *CoNLL'2005*.
Charniak, E., (2000). A maximum-entropy-inspired parser. *NAACL'2000*.
Che, W. X., Zhang, M., Liu, T., & Li, S. (2006). A hybrid convolution tree kernel for semantic role labeling. *COLING-ACL'2006 (poster)*.
Collins, M., & Duffy, N. (2001). Convolution kernels for natural language. *NIPS'2001*.
Culotta, A., & Sorensen, J. (2004). Dependency tree kernels for relation extraction. *ACL'2004*.
Gildea, D., & Jurafsky, D. (2002). Automatic labeling of semantic roles. *Computational Linguistics, 28*(3), 245–288.
Hajic, J., Ciaramita, M., Johansson, R., et al. (2009). The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. *CoNLL'2009*.
Haussler, D. (1999). Convolution kernels on discrete structures. *Technical report UCSC-CRL-99-10*.
Joachims, T. (1998). Text categorization with support vector machine: learning with many relevant features. *ECML'1998*.
Kashima, H. (2007). *Machine learning approaches for structured data*. PhD dissertation, Kyoto University, Japan.
Kashima, H., & Koyanagi, T. (2002). Kernels for semi-structured data. *ICML'2002*.
Kipper, K., Korhonen, A., Ryant, N., & Palmer, M. (2008). A large-scale classification of English verbs. *Language Resources and Evaluation, 42*(1), 21–40.
Liu, T., Che, W. X., & Li, S. (2007). Semantic role labeling with maximum entropy classifier. *Journal of Software, 18*(3), 565–573 (in Chinese).
Marcus, M. P., Marcinkiewicz, M. A., & Santorini, B. (1993). Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics, 19*(2), 313–330.
Moschitti, A. (2004). A study on convolution kernels for shallow statistic parsing. *ACL'2004*.
Moschitti, A. (2006). Efficient convolution kernels for dependency and constituent syntactic trees. *ECML'2006*.
Moschitti, A., Pighin, D., & Basili, R. (2008). Tree kernels for semantic role labeling. *Computational Linguistics, 34*(2), 193–224.
Narayanan, S., & Harbabagiu, S. (2004), Question answering based on semantic structures. *COLING'2004*.
Palmer, M., Gildea, D., & Kingsbury, P. (2005). The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics, 31*(1), 71–105.
Ponzetto, S. P., & Strube, M., (2006). Exploiting semantic role labeling, wordnet and wikipedia for coreference resolution. *HLT-NAACL'2006*.
Pradhan, S., Ward, W., Hacioglu, K., Martin, J., & Jurafsky, D. (2004). Shallow semantic parsing using support vector machines. *HLT/NAACL'2004*.
Pradhan, S., Hacioglu, K., Krugler, V., Ward, W., Martin, J. H., & Jurafsky, D. (2005). Support vector learning for semantic argument classification. *Machine Learning, 60*(1), 11–39.
Punyakanok, V., Roth, D., & Yih, W. T. (2005). The necessity of syntactic parsing for semantic role labeling. *IJCAI'2005*.
Shen, L. B., Sarkar, A., & Joshi, A. K. (2003). Using LTAG based features in parse reranking. *EMNLP'2003*.
Surdeanu, M., Harabagiu, S., Williams, J., & Aarseth, P. (2003). Using predicate-argument structures for information extraction. *ACL'2003*.
Surdeanu, M., Johansson, R., Meyers, A., Màrquez, L., & Nivre, J. (2008). The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. *CoNLL'2008*.
Vapnik, V. N. (1998). *Statistical learning theory*. Wiley.
Zelenko, D., Aone, C., & Richardella, A. (2003). Kernel methods for relation extraction. *Journal of Machine Learning Research, 3*, 1083–1106.
Zhang, M., Zhang, J., Su, J., & Zhou, G. D. (2006). A composite kernel to extract relations between entities with both flat and structured features. *COLING-ACL'2006*.
Zhang, M., Che, W. X., Aw, A. T., Tan, C. L., Zhou, G. D., Liu, T., et al. (2007). A grammar-driven convolution tree kernel for semantic role classification. *ACL'2007*.
Zhou, G. D., Zhang, M., Ji, D. H., & Zhu, Q. M. (2007). Tree kernel-based relation extraction with context-sensitive structured parse tree information. *EMNLP-CoNLL'2007*.