Available online at www.sciencedirect.com

ScienceDirect

Information Processing and Management 44 (2008) 687–701

INFORMATION PROCESSING & MANAGEMENT

www.elsevier.com/locate/infoproman

Exploring syntactic structured features over parse trees for relation extraction using kernel methods ☆

Zhang Min [a,*], Zhou GuoDong [a,b], Aw Aiti [a]

[a] *Institute for Infocomm Research, 21 Heng Mui Keng Terrace, Singapore 119613, Singapore*
[b] *School of Computer Science and Technology, Soochow University, 215006, China*

Received 28 November 2006; received in revised form 28 July 2007; accepted 30 July 2007
Available online 11 September 2007

Abstract

Extracting semantic relationships between entities from text documents is challenging in information extraction and important for deep information processing and management. This paper proposes to use the convolution kernel over parse trees together with support vector machines to model syntactic structured information for relation extraction. Compared with linear kernels, tree kernels can effectively explore implicitly huge syntactic structured features embedded in a parse tree. Our study reveals that the syntactic structured features embedded in a parse tree are very effective in relation extraction and can be well captured by the convolution tree kernel. Evaluation on the ACE benchmark corpora shows that using the convolution tree kernel only can achieve comparable performance with previous best-reported feature-based methods. It also shows that our method significantly outperforms previous two dependency tree kernels for relation extraction. Moreover, this paper proposes a composite kernel for relation extraction by combining the convolution tree kernel with a simple linear kernel. Our study reveals that the composite kernel can effectively capture both flat and structured features without extensive feature engineering, and easily scale to include more features. Evaluation on the ACE benchmark corpora shows that the composite kernel outperforms previous best-reported methods in relation extraction.
© 2007 Elsevier Ltd. All rights reserved.

Keywords: Information extraction; Relation extraction; Syntactic structured features; Convolution tree kernel; Composite kernel

1. Introduction

Relation extraction is to find various predefined semantic relations between pairs of entities in text. The research on relation extraction has been promoted by the Message Understanding Conferences (MUCs)

☆ Some preliminary results have been reported in COLING-ACL'2006 (Zhang, Zhang, Su, & Zhou, 2006).
* Corresponding author. Tel.: +65 68748205; fax: +65 67755014.
E-mail addresses: mzhang@i2r.a-star.edu.sg (M. Zhang), zhougd@i2r.a-star.edu.sg, gdzhou@suda.edu.cn (G. Zhou), aaiti@i2r.a-star.edu.sg (A. Aw).

0306-4573/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.ipm.2007.07.013

(MUC, 1987–1998) and the NIST Automatic Content Extraction (ACE) program (ACE, 2002–2006). According to the ACE Program, an *entity* is an object or 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 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.Commercial). Extraction of such kinds of semantic relations between entities can be very useful for 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 country "the United States".

Much research work has been done for relation extraction. Prior feature-based methods for this task (Kambhatla, 2004; Zhou, Su, Zhang, & Zhang, 2005) employed a large amount of diverse linguistic features, varying from lexical knowledge, entity mention information to syntactic parse trees, dependency trees and semantic features. Since a parse tree contains rich syntactic structured information, in principle, the structured features extracted from a parse tree should contribute much to performance improvement in relation extraction. However it is reported (Kambhatla, 2004; Zhou et al., 2005) that syntactic structured features contributes little to performance improvement. This may be mainly due to the fact that the hierarchical structured information in a parse tree is hard to be explicitly represented by a vector of linear features. As an alternative, kernel methods (Collins & Duffy, 2001) provide an elegant solution to explore implicitly structured features by directly computing the similarity between two trees. However, the sole two reported dependency tree kernels in relation extraction on the ACE corpora (Bunescu & Mooney, 2005; Culotta & Sorensen, 2004) showed much lower performance than the feature-based methods. One may ask: Are the structured features in syntactic parse trees useful in relation extraction? Can tree kernel methods effectively capture the structured features as well as other various flat features that have been proven useful in the feature-based methods?

In this paper, we demonstrate the effectiveness of the syntactic structured features in relation extraction and study how to capture such features via a convolution tree kernel (Collins & Duffy, 2001) together with support vector machines (SVM) (Vapnik, 1998). We also study how to select the proper feature space (e.g. the set of sub-trees to represent relation instances) to optimize the system performance. The experimental results show that the convolution tree kernel can achieve comparable performance with previous best-reported feature-based methods. It also shows that our kernel method significantly outperforms previous two dependency tree kernels (Bunescu & Mooney, 2005; Culotta & Sorensen, 2004) for relation extraction.

Moreover, this paper proposes a composite kernel for relation extraction by combining the convolution tree kernel with a simple linear kernel. Our study demonstrates that the composite kernel is very effective in relation extraction. It also shows that, without extensive feature engineering, the composite kernel can not only capture most of the flat features used in previous work but also exploit the useful syntactic structured features effectively. An advantage of our method is that the composite kernel can easily cover more types of knowledge by introducing more kernels. Evaluation shows that our method outperforms previous best-reported methods and significantly outperforms previous kernel methods due to its effective exploration of the syntactic structured features.

The rest of the paper is organized as follows. In Section 2, we review previous work. Then, a brief introduction about kernel-based relation extraction is given in Section 3. Section 4 discusses different structured feature spaces over parse trees and the convolution tree kernel for relation extraction while the composite kernel is described in Section 5. Section 6 shows the experimental results while further comparison of our work with related work is presented in Section 7. Finally, we conclude our work in Section 8.

2. Related work

The task of relation extraction was first introduced as a part of the Template Element task in MUC6 and formulated as the Template Relation task in MUC7 (MUC, 1987–1998). Since then, many methods on relation extraction, such as generative models (Miller, Fox, Ramshaw, & Weischedel, 2000, 19871998), feature-based (Kambhatla, 2004; Zhou et al., 2005) and kernel-based (Bunescu & Mooney, 2005; Culotta & Sorensen, 2004; Zelenko, Aone, & Richardella, 2003) have been proposed in the literature.

Miller et al. (2000) address the task of relation extraction from the statistical parsing viewpoint. They augment syntactic full parse trees with semantic information corresponding to entities and relations, and build generative models to integrate various tasks such as POS tagging, named entity recognition, template element extraction and relation extraction. Their results essentially depend on the entire full parse tree and need a large scale of annotated corpus.

For the feature-based methods, Kambhatla (2004) employs Maximum Entropy models to combine diverse lexical, syntactic and semantic features derived from the text in relation extraction. Zhou et al. (2005) explore various features in relation extraction using SVM. They conduct exhaustive experiments to investigate the incorporation and the individual contribution of diverse features. They report that chunking information contributes to most of the performance improvement from the syntactic aspect. The features used in Kambhatla (2004) and Zhou et al. (2005) have to be selected and carefully calibrated manually. Kambhatla (2004) uses the path of non-terminals connecting two entities in a parse tree as the parse tree features while Zhou et al. (2005) introduce additional chunking features to enhance the parse tree features. However, the hierarchical structured information in the parse trees is not well preserved in their parse tree features.

As an alternative to the feature-based methods, kernel methods (Haussler, 1999) 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 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. In relation extraction, typical work using kernel methods includes Zelenko et al. (2003), Culotta and Sorensen (2004) and Bunescu and Mooney (2005).

Zelenko et al. (2003) develop a parse tree kernel for relation extraction. Their tree kernel is recursively defined in a top-down manner, matching nodes from roots to leaf nodes. For each pair of matching nodes, a subsequence kernel on their child nodes is invoked, which matches either contiguous or sparse subsequences of nodes. Culotta and Sorensen (2004) generalize this kernel to estimate the similarity between dependency trees. One may note that their tree kernel requires the matchable nodes must be at the same depth counting from the root node. This is a strong constraint on the matching of syntax so it is not surprising that the model has good precision but very low recall on the ACE corpora. In addition, according to the top-down node-matching mechanism of the kernel, once a node is not matchable with any node in the same layer in another tree, all the sub-trees below this node are discarded even if some of them are matchable to their counterparts in another tree.

Bunescu and Mooney (2005) propose a shortest path dependency kernel in relation extraction. They argue that the information to model a relationship between entities is typically captured by the shortest path between the two entities in the dependency graph. Their kernel is very straightforward. It just sums up the number of common word classes at each position in the two paths. We notice that one issue of this kernel is that they limit the two paths must have the same length, otherwise the kernel similarity score is zero. Therefore, although this kernel shows non-trivial performance improvement than that of Culotta and Sorensen (2004), the constraint makes the two dependency kernels share the similar behavior: good precision but much lower recall on the ACE corpora.

Zhao and Grishman (2005) define a feature-based composite kernel to integrate diverse features for relation extraction. Their kernel displays very good performance on the 2004 version of ACE corpora. Since this is a feature-based kernel, all the features used in the kernel have to be explicitly enumerated. Similar to the feature-based methods, they also represent the tree feature as a link path between two entities. Therefore, we wonder whether their performance improvement is mainly due to the explicit incorporation of diverse linguistic features instead of the kernel method itself. Here, we would classify their method into the feature-based methods since their kernels can be easily represented by the dot products between explicit feature vectors.

The above discussion suggests that the hierarchical structured features in a parse tree may not be fully utilized in previous work, no matter whether feature-based or kernel-based. We believe that the tree structure features could play a more important role than that reported in previous work. Since convolution kernels (Haussler, 1999)[1] aim to capture structured information in terms of sub-structures, which providing a viable alternative to flat features, in this paper, we propose to use a convolution tree kernel (Collins & Duffy, 2001) to explore syntactic structured features in relation extraction. To our knowledge, convolution kernels have not been explored in relation extraction. Furthermore, in order to integrate both flat and structured features for relation extraction, two composite kernels are proposed and studied in this paper.

3. Kernel-based classifiers for relation extraction

In this paper, relation extraction is re-cast as a classification problem using a machine learning algorithm. In training, a classifier machine learning algorithm uses the annotated relation instances to learn a classifier while, in testing, the learned classifier is applied to input instances to determine their relation classes (including non-relation) and thus extract possible relations.

Most learning algorithms rely on feature-based representation of input instances. That is, an annotated instance is transformed into a collection of features $f_1, f_2, \ldots, f_N$, thereby producing an N-dimensional feature vector. However, in many NLP problems, it is computationally infeasible to generate features involving structured information or long-distance dependencies. For example, one cannot enumerate efficiently all the subtree features for a full parse tree.

As an alternative to the feature-based methods, kernel methods (Haussler, 1999) can implicitly explore high-dimensional structured features efficiently using a kernel function. A kernel function is a similarity function satisfying the properties of being symmetric[2] and positive-definite.[3] More precisely, a kernel function K over the instance space X $K: X \times X \rightarrow [0, \infty]$ maps a pair of instances $x, y \in X$ to their similarity score $K(x, y)$. It can be proven that a kernel function can measure the similarity between two input instances by computing implicitly the dot product of certain features in high (or even infinite)-dimensional feature spaces without enumerating all the features (Vapnik, 1998). In this paper, we propose to use the convolution tree kernel (Collins & Duffy, 2001) to model syntactic structured features for relation extraction and further propose two composite kernels to integrate both flat and structured features for relation extraction.

Many classifiers, such as SVM, KNN and voted perceptrons, can be used with kernels by replacing the dot product with a kernel function. In this paper, we select SVM as the classifier since SVM represents the state-of-the-art in the machine learning research community, and there are good implementations of the algorithm available. In our implementation, we use the binary-class SVMLight deleveloped by Joachims (1998). SVM is a supervised machine learning technique motivated by the statistical learning theory (Vapnik, 1998). Based on the structural risk minimization of the statistical learning theory, SVM seeks an optimal separating hyper-plane to divide the training examples into two classes and make decisions based on support vectors that are selected as the only effective instances in the training set. Basically, SVM is a binary classifier. Therefore, we must extend SVM to multi-class (e.g. K) classification for the ACE relation extraction task. For efficiency, we apply the *one vs. others* strategy, which builds K classifiers so as to separate one class from all others, instead of the *pairwise* strategy, which builds $K * (K - 1)/2$ classifiers considering all pairs of classes. The final decision of an instance in the multiple binary classification is determined by the class which has the maximal SVM output.

[1] Convolution kernels were proposed as a concept of kernels for a discrete structure by Haussler (1999) in machine learning study. This framework defines a kernel between input objects by applying convolution "sub-kernels" that are the kernels for the decompositions (parts) of the objects. Convolution kernels are abstract concepts, and the instances of them are determined by the definition of "sub-kernels". The *Tree Kernel* (Collins & Duffy, 2001), *String Subsequence Kernel* (SSK) (Lodhi, Saunders, Shawe-Taylor, Cristianini, & Watkins, 2002) and *Graph Kernel* (HDAG Kernel) (Suzuki, Hirao, Sasaki, & Maeda, 2003) are examples of convolution kernels instances in the NLP field.

[2] A binary function $K(\cdot,\cdot)$ is symmetric (over X), if $\forall\ x, y \in X, K(x, y) = K(y, x)$.

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

4. Tree kernels for relation extraction

In this section, we discuss the convolution tree kernel associated with different implicit relation feature spaces. In Section 4.1, we define seven different relation feature spaces over parse trees. In Section 4.2, we introduce a convolution tree kernel for relation extraction.

4.1. Feature spaces[4]

A relation instance is encapsulated by a parse tree. Thus, it is critical to understand which portion of a parse tree is important in kernel calculation. For this purpose, seven different feature spaces are defined as follows:

(1) *Minimum complete tree (MCT)*: The complete sub-tree rooted by the nearest common ancestor of the two entities under consideration (e.g. T_1 in Fig. 1).
(2) *Path-enclosed tree (PT)*: The smallest common sub-tree including the two entities. In other words, the sub-tree is enclosed by the shortest path linking the two entities in the parse tree (this path is also commonly used as the path tree feature in the feature-based methods) (e.g. T_2 in Fig. 1).
(3) *Chunking tree (CT)*: The base phrase list extracted from the PT by pruning out all the internal structures of the PT while only keeping the root node and the base phrase list for generating the chunking tree (e.g. T_3 in Fig. 1).
(4) *Context-sensitive path tree (CPT)*: The PT extending with the 1st left sibling of the node of entity 1 and the 1st right sibling of the node of entity 2. If the sibling is unavailable, then we move to the parent of current node and repeat the same process until the sibling is available or the root is reached (e.g. T_4 in Fig. 1).
(5) *Context-sensitive chunking tree (CCT)*: The CT extending with the 1st left sibling of the node of entity 1 and the 1st right sibling of the node of entity 2. If the sibling is unavailable, the same process as generating the CPT is applied. Then we do a further pruning process to guarantee that the context structure in the CCT is still a list of base phrases (e.g. T_5 in Fig. 1).
(6) *Flattened PT (FPT)*: The PT with (1) the single in and out arcs of non-terminal nodes (except POS nodes) removed, and (2) the non-terminal nodes having the same syntactic phrase type with their father removed (e.g. T_6 in Fig. 1).
(7) *Flattened CPT (FCPT)*: The CPT with (1) the single in and out arcs of non-terminal nodes (except POS nodes) removed, and (2) the non-terminal nodes having the same syntactic phrase type with their father removed (e.g. T_7 in Fig. 1).

As an example, Fig. 1 illustrates different sub-tree structures for a relation instance in the sentence "*Akyetsu testified he was powerless to stop the merger of an estimated 2000 ethnic Tutsi's in the district of Tawba.*", excerpted from the 2003 version of ACE corpora, where an ACE-defined relation "AT.LOCATED" exists between the entities "*Tutsi's*" (PER) and "*district*" (GPE). We use Charniak's parser (Charniak, 2001) to parse the example sentence. Due to space limitation, we do not show the whole parse tree of the entire sentence here. In Fig. 1,

(1) T_1 is MCT for the relation instance, where the sub-structure circled by a dashed line is PT, which is also shown in T_2 for clarity. The only difference between MCT andPT lies in that MCT does not allow partial

[4] In feature-based methods, the feature space of a parse tree is represented by an explicit vector of *n*-dimensional predefined features extracted from the parse tree. However, in tree kernel-based methods, the similarity between two parse trees are computed directly from the two parse trees by implicitly mapping a parse tree to a feature vector of infinite dimension. For convenience, we use "feature space" to refer to both the explicit vector of *n*-dimensional predefined features in feature-based methods and the implicit feature vector of infinite dimension in tree kernel-based methods. In this paper, we employ a tree kernel-based method and represent a parse tree as an implicit vector of integer counts of each sub-tree type (regardless of its ancestors). For more details, please refer to Sections 4.2 and 7.1.

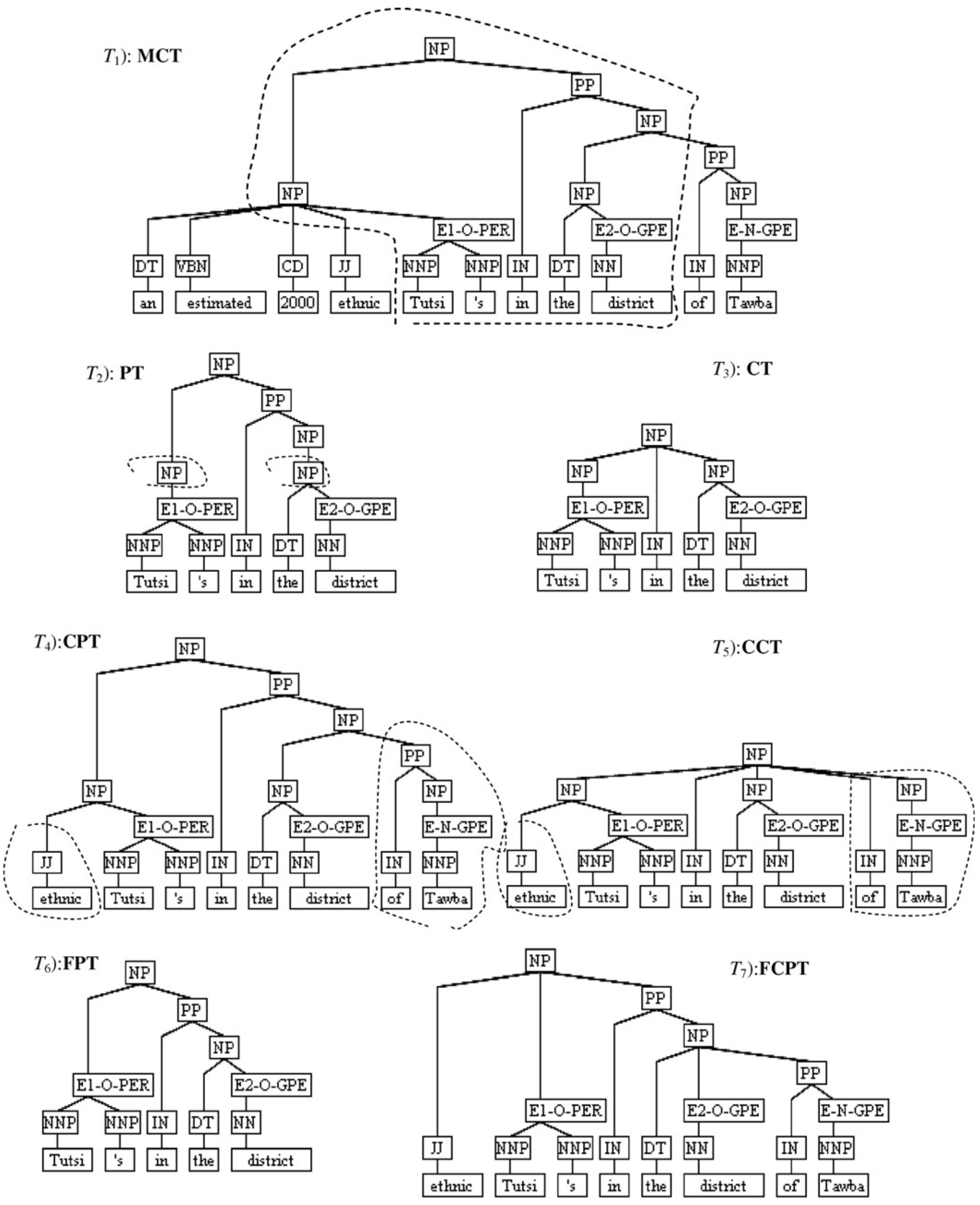

Fig. 1. Feature spaces of the example sentence "... to stop the merger of an estimated 2000 ethnic Tutsi's in the district of Tawba.", where the phrase type "E1-O-PER" denotes that the current phrase is the 1st entity, its entity type is "PERSON" and its mention level is "NOMIAL", and likewise for the other two phrase types "E2-O-GPE" and "E-N-GPE".

production rules. For instance, the most-left two-layer sub-tree [NP [DT ... E1-O-PER]] in T_1 is broken apart in T_2. By comparing the performance of T_1 and T_2, we can evaluate the effect of sub-trees with partial production rules as shown in T_2 and the necessity of keeping the whole left and right context sub-trees as shown in T_1 in relation extraction.

(2) T_3 is CT. By comparing the performance of T_2 and T_3, we want to study whether the chunking information or the hierarchical structure information embedded in a parse tree is more effective for relation extraction.

(3) T_4 is CPT, where the two structures circled by dashed lines are included as the context to T_2 (PT) and make T_4 context-sensitive. This is to evaluate whether the limited context information in CPT can boost performance.
(4) T_5 is CCT, where the additional context structures are also circled by dashed lines. This is to study whether the limited context information in CCT can boost performance.
(5) Two flattened trees as given in T_6 and T_7 are also explored. The two circled nodes in T_2 are removed in the flattened trees T_6 and T_7. This is to study if the eliminated small structures are noisy for relation extraction.

4.2. Convolution tree kernel

Given the various feature spaces discussed in the previous subsection, we now study how to measure the similarity between two trees using convolution tree kernel in these feature spaces.

A convolution kernel (Haussler, 1999) aims to capture structured information in terms of sub-structures. As a specialized convolution kernel, the convolution tree kernel suggested by Collins and Duffy (2001) counts the number of common sub-trees (sub-structures) as the syntactic structured similarity between two parse trees. In their vector representation of a parse tree, a parse tree T is represented by a vector of integer counts of each sub-tree type (regardless of its ancestors):

$$\phi(T) = (\#subtree_1(T), \ldots, \#subtree_i(T), \ldots, \#subtree_n(T)) \tag{1}$$

where $\#subtree_i(T)$ is the occurrence number of the ith sub-tree type ($subtree_i$) in T. Since the number of different sub-trees is exponential with the parse tree size, it is computationally infeasible to directly use the feature vector $\phi(T)$. To solve this computational issue, Collins and Duffy (2001) proposed the following parse tree kernel $K(T_1, T_2)$ to calculate the dot product between the above high-dimensional vectors implicitly

$$\begin{aligned} K(T_1,T_2) &= \langle \phi(T_1), \phi(T_2) \rangle \\ &= \sum_i \#subtree_i(T_1) \cdot \#subtree_i(T_2) \\ &= \sum_i \left(\left(\sum_{n_1 \in N_1} I_{subtree_i}(n_1) \right) \cdot \left(\sum_{n_2 \in N_2} I_{subtree_i}(n_2) \right) \right) \\ &= \sum_{n_1 \in N_1} \sum_{n_2 \in N_2} \Delta(n_1, n_2) \end{aligned} \tag{2}$$

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

$$\Delta(n_1,n_2) = \sum_i I_{subtree_i}(n_1) \cdot I_{subtree_i}(n_2) \tag{3}$$

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

Rule 1: if the productions (CFG rules) at n_1 and n_2 are different, $\Delta(n_1,n_2) = 0$;
Rule 2: else if both n_1 and n_2 are pre-terminals (POS tags), $\Delta(n_1,n_2) = 1 \times \lambda$;
Rule 3: else, $\Delta(n_1,n_2) = \lambda \prod_{j=1}^{nc(n_1)} (1 + \Delta(ch(n_1,j), ch(n_2,j)))$, where $nc(n_1)$ is the child number of n_1, $ch(n,j)$ is the jth child of node n and $\lambda (0 < \lambda < 1)$ is the decay factor in order to make the kernel value less variable with respect to the *subtree* sizes.

The recursive rule (3) holds because given two nodes with the same children, one can construct common sub-trees using these children and common sub-trees of further offspring. The time complexity for computing this kernel is $O(|N_1| \cdot |N_2|)$.

5. Composite kernels for relation extraction

Our proposed composite kernels integrate the convolution tree kernel described as above with a linear entity-based kernel in order to capture both structured and flat features for relation extraction.

The linear entity-based kernel is based on entity-related features as defined in ACE program. For example, the ACE 2003 data defines four kinds of entity features: entity headword, entity type and subtype (only for GPE[5]), and mention type while the ACE 2004 data makes some modifications and introduces a new feature "LDC mention type". Our statistics on the ACE data reveals that the entity features impose a strong constraint on relation types. Therefore, we design a linear kernel to capture explicitly such features:

$$K_L(R_1, R_2) = \sum_{i=1,2} K_E(R_1 \cdot E_i, R_2 \cdot E_i) \tag{4}$$

where R_1 and R_2 stands for two relation instances, E_i means the ith entity of a relation instance, and $K_E(\cdot,\cdot)$ is a simple kernel function over the features of entities:

$$K_E(E_1, E_2) = \sum_{i} C(E_1 \cdot f_i, E_2 \cdot f_i) \tag{5}$$

where f_i represents the ith entity feature, and the function $C(\cdot,\cdot)$ returns 1 if the two feature values are identical and 0 otherwise. $K_E(\cdot,\cdot)$ returns the number of feature values in common of two entities. Eq. (4) is a proper kernel since it simply calculates the dot product of the entity feature vectors.

Given the above linear entity-based kernel, two composite kernels are explored in this paper:

(1) *Linear combination*

$$K_1(R_1, R_2) = \alpha \cdot \widehat{K}_L(R_1, R_2) + (1 - \alpha) \cdot \widehat{K}(T_1, T_2) \tag{6}$$

Here, $\widehat{K}(\cdot,\cdot)$ is the normalized[6] $K(\cdot,\cdot)$ and α is the coefficient. Evaluation on the development set shows that this composite kernel yields best performance when α is set to 0.4.

(2) *Polynomial expansion*

$$K_2(R_1, R_2) = \alpha \cdot \widehat{K}_L^p(R_1, R_2) + (1 - \alpha) \cdot \widehat{K}(T_1, T_2) \tag{7}$$

Here, $\widehat{K}(\cdot,\cdot)$ is the normalized $K(\cdot,\cdot)$, $K^p(\cdot,\cdot)$ 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. Evaluation on the development set shows that this composite kernel yields best performance when α is set to 0.23.[7]

The polynomial expansion aims to explore the entity bi-gram features, esp. the combined features from the first and second entities, respectively. In addition, due to the different scales of the values of the two individual kernels, they are normalized before combination. This can avoid one kernel value being overwhelmed by that of another one.

Since a kernel function set is closed under normalization, polynomial expansion and linear combination (Schölkopf & Smola, 2001), the two composite kernels are also proper kernels.

6. Experimentation

The main aim of our experiment is to verify the effectiveness of using richer syntactic structures and the convolution tree kernel in relation extraction. Moreover, we also evaluate the effectiveness of the two proposed composite kernels.

[5] GPE is a Geo-Political Entity – an entity with land and a government, such as United States of America.

[6] A kernel $K(x,y)$ can be normalized by dividing it by $\sqrt{K(x,x) \cdot K(y,y)}$.

[7] The coefficient α in the two composite kernels are determined by a development set. We just simply select the value of α which generates the best results on the development set. The results are not so sensitive to the different values of α unless its value is less than 0.13/0.08 or more than 0.92/0.86 in the two composite kernels, respectively.

Table 1
Statistics of relation types and subtypes in the ACE 2003 training data (relations marked with an ∗ are symmetric relations)

Type	Subtype	Occurrence frequency
AT(2781)	Based-in	347
	Located	2126
	Residence	308
NEAR(201)PART(1298)	Relative-location*	201
	Part-of	947
	Subsidiary	355
	Other	6
ROLE(4756)	Affiliate-partner	204
	Citizen-of	328
	Client	144
	Founder	26
	General-staff	1331
	Management	1242
	Member	1091
	Owner	232
	Other	158
SOCIAL(827)	Associate*	91
	Grandparent	12
	Other-personal	85
	Other-professional*	339
	Other-relative*	78
	Parent	127
	Sibling*	18
	Spouse*	77

6.1. Experimental setting

We use the English portion of both the ACE 2003 and 2004 corpora from LDC in our experiments. These ACE corpora are gathered from various newspapers, newswire and broadcasts. In the ACE 2003 data (LDC2003T11: 422/97 training/testing documents and LDC2004T09: 252/0 training/testing documents), the training set consists of 674 (422 + 252) documents and 9683 relation instances while the test set consists of 97 documents and 1386 relation instances. The ACE 2003 data defines 5 entity types, 5 major relation types and 24 relation subtypes. Table 1 lists the types and subtypes of relations for the ACE 2003 data, along with their frequencies of occurrence in the training set. It shows that this ACE 2003 data suffers from a small amount of annotated data for a few subtypes such as the subtype “Founder” under the type “ROLE”. It also shows that the ACE 2003 task defines some difficult subtypes such as the subtypes “Based-In”, “Located” and “Residence” under the type “AT”, which are difficult even for human experts to differentiate. The ACE 2004 data (LDC2005T09) contains 451 documents and 5702 relation instances. It redefines 7 entity types, 7 major relation types and 23 subtypes. Since Zhao and Grishman (2005) use a 5-fold cross-validation on a subset of the 2004 data (newswire and broadcast news domains, containing 348 documents and 4400 relation instances), for comparison, we use the same setting for the ACE 2004 data.

Both corpora are parsed using Charniak’s parser (Charniak, 2001). We iterate over all pairs of entity mentions occurring in the same sentence to generate potential relation instances.[8] We explicitly model the argument order of the two mentions involved. For example, when comparing mentions m1 and m2, we distinguish between m1-ROLE.Citizen-Of-m2 and m2-ROLE.Citizen-Of-m1. Note that, in the 2003 data, 6 of these 24 relation subtypes are symmetric: “NEAR.Relative-Location”, “SOCIAL.Associate”, “SOCIAL.-Other-Relative”, “SOCIAL.Other-Professional”, “SOCIAL.Sibling”, and “SOCIAL.Spouse”. In this way, we model relation extraction as a multi-class classification task with 43 ($24 \times 2 - 6 + 1$) classes, two for each

[8] In this paper, we only measure the performance of relation extraction models on “true” mentions with “true” chaining of coreference (i.e. as annotated by LDC annotators). Moreover, we only model explicit relations because of poor inter-annotator agreement in the annotation of implicit relations and their limited number.

relation subtype (except the above 6 symmetric subtypes) and a "NONE" class for the case where the two mentions are not related. For the ACE RDC 2004 task, 6 of these 23 relation subtypes are symmetric: "PHYS.Near", "PER-SOC.Business", "PER-SOC.Family", "PER-SOC.Other", "EMP-ORG.Partner", and "EMP-ORG.Other". In this way, we model relation extraction as a multi-class classification task with 41 ($23 \times 2 - 6 + 1$) classes, two for each relation subtype (except the above 6 symmetric subtypes) and a "NONE" class for the case where the two mentions are not related. As discussed in Section 3, SVM is selected as our kernel classifier. The training parameters are chosen using cross-validation ($C = 2.4$ (SVM); $\lambda = 0.4$(tree kernel)). In our implementation, we use SVMLight (Joachims, 1998) and Tree Kernel Tools (Moschitti, 2004) while Precision (P), Recall (R) and F-measure (F) are adopted to measure the performance.

6.2. Experimental results

In this subsection, we report the experimental results of different kernel setups.

6.2.1. Tree kernel over different feature spaces

In order to better study the impact of the syntactic structured information in a parse tree on relation extraction, we remove the entity-related information from parse trees by replacing the entity-related phrase types ("E1-PER" and so on as shown in Fig. 1 (Fig. 1 is the original entity-inclusive tree)) with "NP". Table 2 compares the performance of seven tree kernel setups on the ACE 2003 data using the tree structure information only. It shows that:

- Overall, the seven different feature spaces are all somewhat effective for relation extraction. This suggests that syntactic structured information has good predication power in relation extraction and the syntactic structured information can be well captured by the tree kernel.
- MCT performs much worse than the others. The reasons may be that MCT includes too much left and right contextual information, which may introduce many noisy features and cause over-fitting (high precision and very low recall as shown in Table 2). This suggests that only keeping the complete (not partial) production rules in MCT does harm performance.
- PT achieves best performance. This means that only keeping the portion of a parse tree enclosed by the shortest path between entities can model relations better than all others. This may be due to that most significant information is with PT and including contextual information may introduce too much noise. Although context may include some useful information, it is still an open question on how to utilize correctly such useful information in the tree kernel for relation extraction.
- The performance of using CT drops by 5.8 in F-measure compared with that of using PT. This suggests that the middle and high-level structures beyond chunking are also very useful for relation extraction.
- The two context-sensitive trees (CPT and CCT) show lower performance than the corresponding original PT and CT. In some cases (e.g. in sentence "the merge of company A and company B...", "merge" is the context word), the context information is helpful. However, the effective scope of context is hard to determine given the complexity and variability of natural languages.
- The two flattened trees (FPT and FCPT) perform a little bit worse than the original trees. This suggests that the internal structures represented by single non-terminal nodes are useful for relation extraction.

Table 2
Performance comparison of seven different tree kernel setups over the ACE 2003 five major types using the parse tree structure information only (regardless of any entity-related information)

Instance spaces	P (%)	R (%)	F
Minimum complete tree (MCT)	77.5	38.4	51.3
Path-enclosed tree (PT)	72.8	53.8	61.9
Chunking tree (CT)	75.2	44.8	56.1
Context-sensitive PT (CPT)	75.9	48.6	59.2
Context-sensitive CT (CCT)	78.3	40.8	53.7
Flattened PT (FPT)	72.7	51.7	60.4
Flattened CPT (FCPT)	76.1	47.2	58.2

Table 3
Performance comparison of different kernel setups over the ACE major types of both the 2003 data (the numbers in parentheses) and the 2004 data (the numbers outside parentheses)

PT (with tree structure information only)	P (%)	R (%)	F
Entity kernel only with linear kernel	75.1 (77.2)	42.7 (27.9)	54.4 (41.0)
Entity kernel only with polynomial kernel ($d = 2$)	78.2 (79.5)	43.9 (34.6)	56.2 (48.2)
Tree kernel only	72.5 (72.8)	56.7 (53.8)	63.6 (61.9)
Composite kernel 1 (linear combination)	73.5 (76.3)	67.0 (63.0)	70.1 (69.1)
Composite kernel 2 (polynomial expansion)	76.1 (77.3)	68.4 (65.6)	72.1 (70.9)

Evaluation on the ACE 2004 data also shows that PT achieves best performance (72.5/56.7/63.6 in P/R/F) when using the parse tree structured information only.[9] More evaluations with the entity type and order information incorporated into tree nodes ("E1-PER", "E2-PER" and "E-GPE" as shown in Fig. 1) also show that PT performs best with 76.1/62.6/68.7 in P/R/F on the 2003 data and 74.1/62.4/67.7 in P/R/F on the 2004 data. We note that the performance improvement by incorporating the entity info into tree nodes is significant. This suggests that the simple entity information is very useful in relation extraction.

6.2.2. Composite kernels

Table 3 compares the performance of different kernel setups on the ACE major types. It clearly shows that:

- The composite kernels achieve significant performance improvement over the two individual kernels. This indicates that the flat and the structured features are complementary and the composite kernels can well integrate them.
- The composite kernel via the polynomial expansion outperforms the one via the linear combination by ~2 in F-measure (72.1 *vs.* 70.1 and 70.9 *vs.* 69.1). It suggests that the bi-gram features are very useful.
- The entity features are quite useful, which can achieve F-measures of 54.4/41.0 (56.2/48.2 with polynomial expansion $d = 2$) alone and can boost the performance largely by ~7 (70.1–63.2/69.1–61.9) in F-measure when combining with the tree kernel. Table 7 compares the performance of different *d*'s when using the entity kernel only. It shows that $d = 2$ achieves best performance while $d = 4$ performs even worse than $d = 1$. Another interesting observation is that precision increases and recall drops when d increases. This means that the *n*-gram entity features may lead to over-fitting when $d > 2$.
- It is interesting that the ACE 2004 data shows consistent better performance on all setups than the 2003 data although the ACE 2003 data is two times larger than the ACE 2004 data. This may be due to two reasons: (1) The ACE 2004 data defines two new entity types and redefines the relation types and subtypes in order to reduce the inconsistency between LDC annotators; (2) More importantly, the ACE 2004 data defines 43 entity subtypes while there are only 3 subtypes in the 2003 data. For example, the entity kernel using the polynomial kernel performs significantly better by 8.0 (56.2–48.2) in F-measure on the subtypes of the 2004 data than that on the 2003 data.
- Our composite kernel can achieve 77.3/65.6/70.9 and 76.1/68.4/72.1 in P/R/F over the ACE 2003 and 2004 major types, respectively.

6.2.3. Performance comparison

Tables 4 and 5 compare our method with previous work on the ACE 2003 and 2004 data, respectively. They show that our method outperforms previous methods and significantly outperforms previous two dependency kernels.[10] This may be due to two reasons: (1) the dependency tree (Culotta & Sorensen, 2004) and the shortest

[9] Our further experiments with different sizes of training data also show that the PT consistently outperform other relation instance spaces represented by different scopes of sub-trees.

[10] Bunescu and Mooney (2005) only use the corpus LDC2003T11 (422/97 training/testing documents) while we use both LDC2003T11 and LDC2004T09 (674/97 training/testing documents). Culotta and Sorensen (2004) use a generic ACE corpus including about 800 documents (no corpus version is specified). Since the experimental corpora are in different sizes and versions, strictly speaking, it is not ready to compare these methods exactly and fairly. Therefore, Table 4 is only for reference purpose. We hope that we can get a few clues from this table.

Table 4
Performance comparison of difference methods on the ACE 2003/2003 data over both 5 major types (the numbers outside parentheses) and 24 subtypes (the numbers in parentheses)

Methods (2002/2003 data)	P (%)	R (%)	F
Ours: composite kernel 2 (polynomial expansion)	77.3 (64.9)	65.6 (51.2)	70.9 (57.2)
Zhou et al. (2005): feature-based SVM	77.2 (63.1)	60.7 (49.5)	68.0 (55.5)
Kambhatla (2004): feature-based MaxEnt	(–) (63.5)	(–) (45.2)	(–) (52.8)
Ours: tree kernel with entity information at node	76.1 (62.4)	62.6 (48.5)	68.7 (54.6)
Bunescu and Mooney (2005): shortest path dependency kernel	65.5 (–)	43.8 (–)	52.5 (–)
Culotta and Sorensen (2004): dependency kernel	67.1 (–)	35.0 (–)	45.8 (–)

Table 5
Performance comparison on the ACE 2004 data over both 7 major types (the numbers outside parentheses) and 23 subtypes (the numbers in parentheses)

Methods (2004 data)	P (%)	R (%)	F
Ours: composite kernel 2 (polynomial expansion)	76.1 (68.6)	68.4 (59.3)	72.1 (63.6)
Zhao and Grishman (2005): feature-based kernel	69.2 (–)	70.5 (–)	70.4 (–)

path (Bunescu & Mooney, 2005) lack internal hierarchical structured information, so their corresponding kernels can only carry out node-matching directly over the nodes with word tokens; (2) the parse tree kernel has less constraints. That is, it is not restricted by the two constraints of the two dependency kernels (identical layer and ancestors for the matchable nodes and identical length of two shortest paths, as discussed in Section 2).

The above experiments verify the effectiveness of our kernel methods for relation extraction. They suggest that the parse tree kernel can effectively explore the hierarchical structured features that are critical for relation extraction.

6.2.4. Error analysis

Table 6 reports the error distribution of the polynomial composite kernel over the major types on the ACE data. It shows that 83.5% (198 + 115/198 + 115 + 62)/85. 8% (416 + 171/416 + 171 + 96) of the errors result from relation detection and only 16.5%/14.2% of the errors result from relation characterization. This may be due to data imbalance and sparseness issues since we find that the negative samples are eight times more than the positive samples in the training set. Nevertheless, it clearly directs our future work.

Table 6
Error distribution of major types on both the 2003 and 2004 data for the composite kernel using polynomial expansion

Error types	# of error instances	
	2004 data	2003 data
False negative	198	416
False positive	115	171
Cross-type	62	96

Table 7
Performance comparison of different entity kernel setups (different d's) over the ACE 2003 major types

Entity kernel (entity-related features)	P (%)	R (%)	F
Entity kernel only (the same polynomial expansion when $d = 1$)	77.2	27.9	41.0
Entity kernel with polynomial expansion ($d = 2$)	79.5	34.6	48.2
Entity kernel with polynomial expansion ($d = 3$)	82.7	30.1	44.1
Entity kernel with polynomial expansion ($d = 4$)	86.1	22.8	36.1

7. Discussion

In this section, we compare our method with previous work from feature engineering viewpoint and report some other observations and issues in our experiments.

7.1. Comparison with feature-based methods

It would be interesting to review the difference between our convolution tree kernel-based method and feature-based methods. The basic difference between them lies in the relation instance representation (parse tree *vs*. feature vector) and the similarity calculation mechanism (kernel function *vs*. dot product). A relation instance in our method is represented as a parse tree while it is represented as a vector of features in the feature-based methods. Moreover, our method estimates the similarity between two relation instances by only counting the number of common sub-structures while the feature-based methods calculate the dot product between the feature vectors directly. The main difference is different feature spaces. Regarding the parse tree features, our method implicitly represents a parse tree by a vector of integer counts of each sub-tree type, i.e., we consider the entire sub-tree types and their occurring frequencies. In this way, the parse tree-related features (the *path* features and the *chunking* features) used in the feature-based methods can be embedded (as a subset) in our feature space. Moreover, the in-between word features and the entity-related features used in the feature-based methods can be also captured by the tree kernel and the linear entity-based kernel, respectively. Therefore our method has the potential of effectively capturing not only most of previous flat features but also useful syntactic structured features.

7.2. Comparison with previous kernel-based methods

It is also worth comparing our method with previous kernel-based methods. Since our method only counts the occurrence of each sub-tree without considering the layer and the ancestors of the root node of the sub-tree, our method is not limited by the constraints (identical layer and ancestors for the matchable nodes, as discussed in Section 2) in Culotta and Sorensen (2004). Moreover, the difference between our method and Bunescu and Mooney (2005) is that their kernel is defined on the shortest *path* between two entities instead of the entire sub-trees. However, the *path* does not maintain the hierarchical structured information. In addition, their kernel requires that the two paths should have the same length, which is proven empirically too strict. Finally, compared to Zhao and Grishman's kernel, our method directly uses the original representation of a parse tree while they flatten a parse tree into a *link* and a *path* instead.

7.3. Computational issue

The recursively defined convolution tree kernel-based method is much slower compared to feature-based methods. In this paper, the computational issue is solved in three ways. First, the inclusion of the linear entity-based kernel makes the composite kernel converge fast. Furthermore, we find that the small portion (PT) of a full parse tree can effectively represent a relation instance. This significantly improves the speed. Finally, the parse tree kernel requires exact match between two sub-trees, which normally does not occur very frequently. Collins and Duffy (2001) report that in practice, running time for the parse tree kernel is more close to linear $O(|N_1| + |N_2|)$, rather than $O(|N_1| * |N_2|)$. As a result, using the PC with Intel P4 3.0G CPU and 2G RAM, our system only takes about 110 min and 30 min to do training on the ACE 2003 ($\sim$77k training instances) and 2004 ($\sim$33k training instances) data, respectively.

7.4. Further improvement

One of the potential problems in the parse tree kernel is that it carries out exact matches between sub-trees, so that this kernel fails to handle sparse phrases (i.e. "a car" *vs*. "a red car") and near-synonymic grammar tags (for example, the variations of a verb (i.e. go, went, gone)). To some degree, it could possibly lead to over-fitting and compromise the performance. However, the above issues can be handled by allowing grammar-driven partial rule matching and other approximate matching mechanisms in the parse tree kernel calculation.

Finally, it is worth noting that, by introducing more individual kernels, our method can easily scale to cover more features from a multitude of sources (e.g. Wordnet, gazetteers, etc.) that can be brought to bear on the task of relation extraction. In addition, we can also easily implement the feature weighting scheme by adjusting Eq. (5) and the rule (2) in calculating $\Delta(n_1, n_2)$ (see Sections 5 and 4.2).

8. Conclusion and future work

Kernel functions have nice properties. In this paper, we explore the syntactic structured features using convolution kernels over parse trees in relation extraction. Evaluation shows that: (1) the relations between entities can be well represented by carefully calibrating effective portions of parse trees; (2) the hierarchical structured features embedded in a parse tree are particularly effective in relation extraction; (3) the convolution tree kernel can effectively capture the syntactic structured features in relation extraction. Moreover, we have designed a composite kernel in relation extraction to resolve the inherent computational problem in tree kernels. Benefiting from the nice properties of the kernel methods, the composite kernel could well explore and combine the flat features and the structured syntactic features, and therefore outperforms previous best-reported feature-based methods on the ACE corpora.

To our knowledge, this is the first research to demonstrate that, without extensive feature engineering, an individual tree kernel can achieve comparable performance with the feature-based methods. This shows that the syntactic features embedded in a parse tree are particularly useful in relation extraction and which can be well captured by the parse tree kernel. In addition, we find that the proper relation instance representation (i.e. selecting effective portions of parse trees in kernel calculations) is very important for relation extraction.

The immediate extension of our work is to improve the accuracy of relation detection. We may adopt a two-step strategy (Culotta & Sorensen, 2004) to model separately the relation detection and characterization issues. We may integrate more features (such as head words or WordNet semantics) into nodes of parse trees. We can also benefit from machine learning algorithms to study how to solve the data imbalance and sparseness issues from the learning algorithm viewpoint. In the future, we would like to test our algorithm on the other version of the ACE corpora and to develop fast algorithm (Vishwanathan & Smola, 2002) to speed up the training and testing process of convolution kernels. Moreover, we will also want to design a more flexible tree kernel for more accurate similarity measure between parse trees.

Acknowledgements

We would like to thank Dr. Alessandro Moschitti for his help in using his Tree Kernel Toolkits and fine-tuning the system. This research (for ZHOU GuoDong only) 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.

References

ACE (2002–2006). *The Automatic Content Extraction (ACE) projects*. Available from http://www.ldc.upenn.edu/Projects/ACE/.

Bunescu, R. C., & Mooney, R. J. (2005). *A shortest path dependency kernel for relation extraction*. EMNLP-2005, Vancouver, BC (pp. 724–731)

Charniak, E. (2001). *Immediate-head parsing for language models*. ACL-2001, Toulouse, France (pp 129–137).

Collins, M., & Duffy, N. (2001). *Convolution kernels for natural language*. NIPS-2001, Cambridge, MA (pp. 625–632).

Culotta, A., & Sorensen, J. (2004). *Dependency tree kernel for relation extraction*. ACL-2004, Barcelona, Spain (pp. 423–429).

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, Chemnitz, Germany (pp. 137–142).

Kambhatla, N. (2004). *Combining lexical, syntactic and semantic features with Maximum Entropy models for extracting relations*. ACL-2004 (poster), Barcelona, Spain (pp. 178–181).

Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., & Watkins, C. (2002). Text classification using string kernel. *Journal of Machine Learning Research, 2002*(2), 419–444.

Miller, S., Fox, H., Ramshaw, L., & Weischedel, R. (2000). *A novel use of statistical parsing to extract information from text*. ANLP'2000, Seattle, USA (pp. 226–233).

Moschitti, A. (2004). *A study on convolution kernels for shallow semantic parsing*. ACL-2004, Barcelona, Spain (pp. 335–342).

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

Schölkopf, B., & Smola, A. J. (2001). *Learning with kernels: SVM, regularization, optimization and beyond*. Cambridge, MA: MIT Press, pp. 407–423.

Suzuki, J., Hirao, T., Sasaki, Y., & Maeda, E., (2003). *Hierarchical directed acyclic graph kernel: Methods for structured natural language data*. ACL-2003, Sapporo, Japan (pp. 33–40).

Vapnik, V. (1998). *Statistical learning theory*. Chichester, GB: Whiley.

Vishwanathan, S. V. N., & Smola, A. J. (2002). *Fast kernels for string and tree matching*. NIPS-2002, Vancouver, Canada (pp. 569–576).

Zhao, S., & Grishman, R. (2005). *Extracting relations with integrated information using kernel methods*. ACL-2005, Ann Arbor, USA (pp. 419–426).

Zelenko, D., Aone, C., & Richardella, A. (2003). Kernel methods for relation extraction. *Journal of Machine Learning Research, 2003*(2), 1083–1106.

Zhou, G., Su, J., Zhang, J., & Zhang, M. (2005). *Exploring various knowledge in relation extraction*. ACL-2005, Ann Arbor, USA (pp. 427–434).

Zhang, M., Zhang, J., Su, J., & Zhou, G. (2006). *A composite kernel to extract relations between entities with both flat and structured features*. COLING-ACL-2006, Sydney, Australia (pp. 825–832).

Zhang Min received his B.Sc., M.Sc. and Ph.D. from Harbin Institute of Technology in 1991, 1994 and 1997, respectively. He joined the Institute for Infocomm Research, Singapore in 2003 and currently is a senior research fellow at the institute.

Zhou GuoDong received his B.Sc., M.Sc. and Ph.D. from XI'AN Jiaotong Univ. in 1989, Shanghai Jiaotong University in 1992 and National University of Singapore in 1999, respectively. He joined the Institute for Infocomm Research, Singapore in 1999 and had been a scientist at the institute until August 2006. Currently, he is a professor at the School of Computer Science and Technology, Soochow University, Suzhou, China.

Aw Aiti received her B.Sc. from National Univ. of Singapore in 1987. She joined the Institute for Infocomm Research, Singapore in 1997 and currently is a research manager at the institute.