Relation Extraction Using Convolution Tree Kernel Expanded with Entity Features*

Longhua Qian, Guodong Zhou, Qiaomin Zhu, Peide Qian

School of Computer Science and Technology, Soochow University.
215006, Suzhou, China

{qianlonghua, gdzhou, qmzhu, pdqian}@suda.edu.cn

Abstract. This paper proposes a convolution tree kernel-based approach for relation extraction where the parse tree is expanded with entity features such as entity type, subtype, and mention level etc. Our study indicates that not only can our method effectively capture both syntactic structure and entity information of relation instances, but also can avoid the difficulty with tuning the parameters in composite kernels. We also demonstrate that predicate verb information can be used to further improve the performance, though its enhancement is limited. Evaluation on the ACE2004 benchmark corpus shows that our system slightly outperforms both the previous best-reported feature-based and kernel-based systems.

Keywords: Information Extraction; Kernel-based Relation Extraction; Support Vector Machines.

1. Introduction

Information extraction is an important research sub-field in natural language processing (NLP) which aims to identify relevant information from large amount of text documents in digital archives and the WWW. Information extraction subsumes three main tasks, including Entity Detection and Tracking (EDT), Relation Detection and Characterization (RDC), and Event Detection and Characterization (EDC).

This paper will focus on the ACE RDC task[1] and employ kernel method to extract semantic relationships between named entity pairs. Many feature-based approaches transform relation instances into feature vectors of high dimension, and compute the inner dot product between these feature vectors. Current research (Kambhatla 2004, Zhao et al 2005, Zhou et al. 2005, Wang et al. 2006) shows that it is very difficult to extract new effective features from relation examples. Kernel methods are non-parametric estimation techniques that computer a kernel function between data instances. By avoiding transforming data examples into feature vectors, kernel methods can implicitly explore much larger feature space than could be searched by a

* This research is supported by Project 60673041 under the National Natural Science Foundation of China and Project 2006AA01Z147 under the "863" National High-Tech Research and Development of China. We would also like to thank Dr. Alessando Moschitti for his great help in using his Tree Kernel Toolkits, including binary package and source codes.

[1] http://www.ldc.upenn.edu/Projects/ACE/.

feature-based approach. Thereafter, kernel methods especially on discrete structures (Haussler 1999) attract more and more attentions in relation extraction as well as other fields in NLP.

Prior work on kernel methods for relation extraction includes Zelenko et al. (2003), Culotta and Sorensen (2004), Bunescu and Mooney (2005). Due to strong constraints that matching nodes be at the same layer and in the identical path starting from the roots to the current nodes, their kernels achieve good precision but much lower recall on the ACE2003 corpus. Zhang et al. (2006) proposed a composite kernel that consists of two individual kernels: an entity kernel that allows for entity-related features and a convolution parse tree kernel that models syntactic information of relation examples. However, their method needs to manually tune parameters in composite kernels that are often difficult to determine.

This paper describes an expanded convolution parse tree kernel to incorporate entity information into syntactic structure of relation examples. Similar to Zhang et al. (2006), we employ a convolution parse tree kernel in order to model syntactic structures. Different from their method, we use the convolution parse tree kernel expanded with entity information other than a composite kernel. One of our motivations is to capture syntactic and semantic information in a single parse tree for further graceful refinement, the other is that we can avoid the difficulty with tuning parameters in composite kernels. Evaluation on the ACE2004 corpus shows that our method slightly outperforms the previous feature-base and kernel-based methods.

The rest of the paper is organized as follows. First, we present our expanded convolution tree kernel in Section 2. Then, Section 3 reports the experimental setting and results. Finally, we conclude our work with some general observations and indicate future work in Section 4.

2. Expanded Tree Kernel

In this section, we describe the expanded convolution parse tree kernel and demonstrate how entity information can be incorporated into the parse tree.

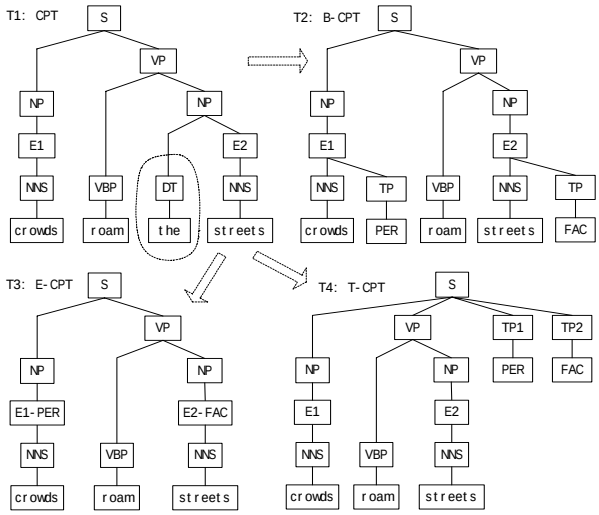

Figure 1: Different representations of a relation instance in the example sentence "in many cities, angry crowds roam the streets.", which is excerpted from the ACE2004 corpus, where a relation "PHSY.Located" holds between the first entity "crowds"(PER) and the second entity "streets" (FAC).

We employ the same convolution tree kernel used by Collins and Duffy (2001), Moschitti (2004) and Zhang et al. (2006). This convolution tree kernel counts the number of subtrees that have similar productions on every node between two parse trees. However, the kernel value will depend greatly on the size of the trees, so we should normalize the kernel.

From ACE definition on relation types and subtypes, we know that entity features impose a strong constraint on relation types. For example, PER-SOC relations describe the relationship between entities of type PER. Zhang et al. (2006) described five cases to extract the portion of parse tree for relation extraction. Their experiments show that PT (Path-enclosed Tree) achieves best performance among those cases, so we begin with PT and then incorporate entity features at different locations as depicted in Figure 1. The four cases is listed as follows:

(1) Compressed Path-enclosed Tree (CPT, T1 in Fig.1): Originated from PT in Zhang et al. (2006), we further make two kinds of compression. One is to prune out the children nodes right before the second entity under the same parent node of NP. The other is to compress the sub-structure like "X-->Y-->Z" into "X-->Z" in the parse trees.
(2) Bottom-attached CPT (B-CPT, T2 in Fig.1): the entity type information is attached to the bottom of the entity node, i.e., two more nodes whose tags are "TP" are added under the first and the second entity nodes respectively.
(3) Entity-attached CPT (E-CPT, T3 in Fig.1): the entity type name is combined with entity order name, e.g. "E1-PER" denotes the first entity whose type is "PER". This case is also explored by Zhang et al. (2006), and we include it here just for the purpose of comparison.
(4) Top-attached CPT (T-CPT, T4 in Fig.1): the entity type information is attached to the top node of the parse tree. In order to distinguish between two entities, we use tags "TP1" and "TP2" to represent the first entity type and the second entity type respectively.

From the above four cases, we want to evaluate whether and how the entity information will be useful for relation extraction and in what way we can embed the entity information (especially the location where we attach) in the parse tree in order to achieve the best performance.

3. Experiments

3.1.Experimental Corpus and Setting

We use the ACE RDC 2004 corpus as our experiment data. The ACE RDC 2004 data contains 451 documents and 5702 relation instances. It defines 7 entity types, 7 major relation types and 23 subtypes. The portion of training data we use contains 347 documents, 121K words and 4307 relations. Evaluation of kernel is done on the training data using 5-fold cross-validation. First, the corpus is parsed using Charniak's parser (Charniak, 2001). Then, we iterate over all pairs of entity mentions occurring in the same sentence to generate potential relation instances.

We choose SVM (Vapnik 1998) as the binary classifier, since SVM has achieved the state-of-the-art performances for many classification problems like text categorization (Joachims 1998). For efficiency, we apply the one-against-others approach to convert binary classifier to multi-class classifier. The final decision of a relation instance in the multi-class classification is determined by the classifier which has the maximal SVM output. In our implementation, we use the binary-class SVMLight (Joachims, 1998) and Tree Kernel Tools (Moschitti, 2004). For comparison with the composite kernels (Zhang et.al. 2006), our training parameter C (SVM) and λ (tree kernel) are set to 2.4 and 0.4 respectively.

3.2.Experimental Results

In this section, we present and analyze the experimental results with respect to different settings.

(1) Different instance representations

According to the above discussion, we select CPT with entity order information as our baseline to try to discover whether and how entity information will be effective to relation extraction. In order to reduce training time, we only add major type information into the parse tree. Table 1 compares the performance of seven major types for three different setups in the ACE2004 corpus using expanded convolution tree kernel. It shows that:

- Using convolution parse tree kernel only embedded with entity order information achieves the performance of 67.8%/52.3%/59.0 in precision/recall/F-measure. This indicates that convolution parse tree kernel is somewhat effective for relation extraction.
- Compared with CPT, other three setups B-CPT, E-CPT and T-CPT improve the F-measure by 8.5/10.1/10.5 units respectively due to the increase both in precision and recall. This shows that entity major type information incorporated into the parse tree of relation instances produces significant improvement for relation extraction. This further suggests that our parse tree kernel can effectively capture both the entity information and the structured syntactic information of relation examples.
- Among the three different instance representations except CPT, the T-CPT (highlighted in bold font) achieves slightly better performance of 2.0/0.4 units in F-measure than the other two representations B-CPT and E-CPT respectively. This may be due to the following reason. From the definition of the convolution parse tree kernel, we introduce a decay factor λ (set to 0.4 here) to make the kernel less dependent on the tree size. However, this factor also decreases the contribution of the entity information on the kernel when they are attached to the bottom of the entity.

Table 1: Performance of seven major types for four different kernel setups in the ACE2004 corpus using expanded convolution tree kernel.

	P	R	F
CPT	67.8	52.3	59.0
B-CPT	75.1	61.4	67.5
E-CPT	76.4	63.1	69.1
T-CPT	**76.0**	**64.0**	**69.5**

(2) Different entity features

In addition to entity type, there are many other entity features about an entity, e.g. subtype, mention level, entity class etc. Do they have different contributions to relation extraction? We will answer this question in the following.

Table 2: Contribution of different entity features over seven major types in the ACE2004 corpus using the above T-CPT kernel. The asterisk on the upper right of the feature means this entity feature can greatly improve the performance while the minus sign means the entity feature doesn't increase the performance and should be removed from the feature set in the next round.

	P	R	F
CPT	67.8	52.3	59.0
+major type*	76.0	64.0	69.5
+subtype*	77.6	64.9	70.7
+mention level*	**79.0**	**66.4**	**72.2**
+entity class$^{(-)}$	79.1	66.2	72.1
+GPE role$^{(-)}$	79.1	66.5	72.2
+head word$^{(-)}$	80.1	64.7	71.6
+LDC type$^{(-)}$	78.9	65.9	71.8
+predicate base	***79.2***	***67.4***	***72.8***

Table 2 reports the contribution of different entity features over seven major types in the ACE2004 corpus using the above T-CPT kernel. It indicates that our system achieves the best performance of 79.0%/66.4%/72.2 in precision/recall/F-measure when combining some of the

entity features. In order to measure the contribution of different entity features we add them one by one in the decreasing order of their potential importance. It also shows:

- Entity type feature is very effective for relation extraction and it increases precision/recall/F-measure by 8.2%/11.7%/10.5 units respectively.
- Entity subtype feature improves the F-measure by 1.2 units. This further shows that gracefully defined entity type and subtype features in the ACE2004 corpus contribute to most of the performance improvement among all entity features.
- Mention level feature is also useful and increases the F-measure by 1.5 units while both entity class and GPE role feature are futile because they don't lead to any improvement in F-measure.
- Other two entity features (i.e. "head word", "LDC mention type"), however, both decrease the performance by 0.6/0.4 units in F-measure respectively. This suggests that both of these features can't differentiate relation types from each other and their incorporations into parse tree make relation extraction even more difficult.
- In the last experiment (highlighted in bold and italic font) we add the base form of the predicate verb nearest to the second entity mention. Although it only improves the F-measure by 0.6 units largely due to the increase in recall, it indicates that moving verbs from the bottom to the top of the parse tree is helpful to relation extraction. This also suggests that constructing a parse tree that contains all necessary features and is designed specifically for relation extraction is very promising.

(3) Different relation lexical condition

In ACE vocabulary, relation lexical condition indicates the syntactic structure where the entity pair relates to each other. There are five relation lexical conditions in the ACE2004 corpus, i.e. "Possessive", "Preposition", "PreMod", "Formulaic" and "Verbal". Table 3 separately measures the recall performance of different relation lexical condition on one of the testing sets in the ACE2004 corpus. It also indicates the number of testing instance, correctly classified instances and wrongly classified instances for each condition respectively.

Table 3: Recall of different lexical conditions on the testing data in the ACE2004 corpus

	#Testing Instances	#Correct Instances	#Error Instances	Recall
Possessive	158	135	23	85.4
Preposition	215	146	69	69.3
PreMod	250	199	51	79.6
Formulaic	71	55	16	77.5
Verbal	174	42	132	24.1

This table shows:

- The recall performance is best in the condition "Possessive". This may be largely due to consistency of syntactic structure for this condition in the ACE2004 corpus.
- It is somewhat surprising that our system performs worse than we expected in the condition "Formulaic", since we think that there should be several fixed patterns for this condition. The reason may be that there are many syntactic errors in the parse trees produced by Charniak's parser although this parser represents the-start-of-art in parsing.
- Finally our system achieves surprisingly lowest performance in the condition "Verbal" although they occur frequently in the testing data. This may be that the syntactic structure in this condition is diverse and it contains too much noise in this kind of parse tree. It also suggests that much more noise needs to be pruned out from the parse tree while the key relation structure should remain in this condition.

(4) Comparison with recent work

Table 4 compares our system with recent work on the ACE2004 corpus. It shows that our system slightly outperforms recently best-reported systems. Compared with the composite kernel (Zhang et al, 2006), our system further prunes the parse tree and incorporates entity features into the convolution parse tree kernel. It shows that our system achieves higher precision, lower recall and slightly better F-measure than their method. Compared with feature-based systems (Zhou et al, 2006 and Zhao et al, 2005) that incorporate many lexical, syntactic and semantic features, our system improves the F-measure by 1.8/2.5 units over relation types respectively. This suggests that kernel-based systems can promisingly outperform feature-based systems, although much work like performance enhancement and reduction of training speed still needs to be done to further improve the system.

Table 4: Comparison of our system with other best-reported systems in the ACE RDC 2004 corpus using 5-fold cross-validation (Note: * for feature-based)

	Relation Detection			RDC on Types		
	P	R	F	P	R	F
Ours: SVM (expansion kernel)	86.3	73.4	79.3	79.2	67.4	72.8
Zhang et al (2006): SVM (polynomial expansion)	-	-	-	76.1	68.4	72.1
Zhou et al (2005): SVM (polynomial kernel)*	89.0	66.6	76.2	82.8	62.1	71.0
Zhao et al (2005): SVM (composite polynomial)*	-	-	-	69.2	70.5	70.3

4. Conclusion and Future Work

In this paper, we have designed a convolution parse tree kernel expanded with entity features for relation extraction using Support Vector Machines. Evaluation on the ACE2004 corpus shows that the expanded convolution parse tree kernel achieves better performance on relation extraction than recent feature-based and kernel-based systems. This may result from the following reasons: First, syntactic structure information of relation examples is very useful and can be effectively captured by the convolution parse tree kernel, therefore the convolution parse tree alone achieves comparable performance on relation extraction. Second, the expanded convolution parse tree incorporated with entity features significantly improves performance. And the higher we put entity feature node in the parse tree, the better performance we can get. We also discover that entity type feature contributes to most of performances improvement while some other features such as "head word" or "GPE role" conversely decrease the performance. Last, compared with other recent systems, performance enhancement of our system is limited, for many parse errors still exist both in short-distance relations and long-distance relations even though the Charniak's parser we use in our system represents the-state-of-the-art in full parsing. This suggests that the parser needs to be further improved in order to provide more accurate syntactic structure information.

In the future work, we will try to construct a dynamic relation tree to reflect both the syntactic structure and semantic information more accurately. First, we will further prune out the noise from the parse tree according to linguistic knowledge especially for lexical condition "Verbal". Second, more weight will be assigned to discriminative features no matter where they are located (e.g. entity features, predicate verb and preposition etc) to reflect their contributions. Last, we will use semantic resources such as WordNet to compute semantic similarity between terminal words (e.g. noun for entity and verb for predicate respectively) in the parse tree.

References

Bunescu R. C. and R. J. Mooney. 2005. A Shortest Path Dependency Kernel for Relation Extraction. *Proceedings of HLT/EMNLP-2005*, pp. 724-731.

Charniak E. 2001. Intermediate-head Parsing for Language Models. *Proceedings of ACL-2001*, pp. 116-123.

Collins M. and N. Duffy. 2001. Convolution Kernels for Natural Language. *Proceedings of NIPS-14*, pp. 625-632.

Culotta A. and J. Sorensen. 2004. Dependency tree kernels for relation extraction. *Proceedings of ACL-2004*, pp. 423-429.

Haussler D. 1999. Convention kernels on discrete structure. *Technical Report UCS-CRL-99-10*.

Joachims T. 1998. Text categorization with support vector machines: learning with many relevant features. *Proceedings of Europe Conference on Machine Learning (ECML-1998)*, pp. 137-142.

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

Moschitti A. 2004. A Study on Convolution Kernels for Shallow Semantic Parsing. *Proceedings of ACL-2004*, pp. 335-342.

Ting W., L. Yaoyong, B. Kalina, C. Hamish, and W. Ji. 2006. Automatic Extraction of Hierarchical Relations from Text. *Proceedings of the Third European Semantic Web Conference (ESWC 2006), Lecture Notes in Computer Science 4011, Springer.*

Vapnik V. 1998. *Statistic Learning Theory*. Chichester: John Wiley.

Zelenko D., C. Aone and A. Richardella. 2003. Kernel Methods for Relation Extraction. *Journal of Machine Learning Research*, 2(2003), 1083-1106.

Zhang M., J. Zhang, J. Su and G. D. Zhou. 2006. A Composite Kernel to Extract Relations between Entities with both Flat and Structured Features. *Proceedings of ACL-2006*, pp. 825–832.

Zhao S.B. and R. Grisman. 2005. Extracting relations with integrated information using kernel methods. *Proceedings of ACL-2005*, pp. 419-426.

Zhou G. D., J. Su, J. Zhang and M. Zhang. 2005. Exploring various knowledge in relation extraction. *Proceedings of ACL-2005*, pp. 427-434.