

Tree Kernel-based Relation Extraction with Various Entity-Related Features*

Longhua Qian, Guodong Zhou, Qiaomin Zhu, Peide Qian
School of Computer Science and Technology, Soochow University.
Suzhou, China, 215006
{qianlonghua, gdzhou, qmzhu, pdqian}@suda.edu.cn

Abstract

This paper proposes a convolution tree kernel-based approach for relation extraction where parse trees are expanded with various entity-related features, such as entity type, subtype, and mention level. Our study indicates that not only can our method effectively capture both syntactic structure and entity information of relation instances in a single tree kernel, but also can avoid the difficulty with tuning the parameters in composite kernels. Moreover, we apply several linguistic rules to further prune out noisy information from the parse trees representing relation instances. Empirical evaluation on the ACE 2004 benchmark corpus shows that our system works well and achieves promising performance compared with other kernel-based systems.

Keywords

Information Extraction; Relation Extraction; Tree Kernel; Support Vector Machines.

1 Introduction

Information extraction is an active research topic in natural language processing (NLP) which aims to identify relevant information from a large amount of unstructured text documents in digital archives and the WWW.

Information extraction subsumes three main sub-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 task¹, which detects and classifies semantic relationships (usually of predefined types) between pairs of entities detected by the EDT task. For example, the sentence “Microsoft Corp. is based in Redmond, WA” conveys the relation “GPE-AFF.Based” between “Microsoft Corp” (ORG) and “Redmond” (GPE). Relation extraction is also very useful in many advanced NLP applications, such as business intelligence, question answering, text summarization and personal profiling.

* A previous version of this paper has been published in PACLIC21 (Qian et al. 2007).

¹ <http://www ldc.upenn.edu/Projects/ACE/>.

In literature, feature-based approaches dominate the research in relation extraction. They 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 relatively practical to construct an efficient feature-based relation extraction system. However, it is also proven difficult to further improve the performance by extracting new effective features from relation examples. In order to resolve this problem, many researchers turn to kernel methods in exploring more structural information. Kernel methods are non-parametric estimation techniques that compute the similarity between two instances. By avoiding transforming instances into feature vectors, kernel methods can implicitly explore much larger feature space than a feature-based approach. Thereafter, kernel methods, especially on discrete structures (Haussler 1999), has been attracting 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. As the last in the list, Zhou et al. (2007) determines a dynamic context-sensitive tree span for relation extraction by extending the widely used Path-closed Tree (PT) to include necessary context-sensitive information outside PT.

This paper also employs the kernel method to tackle the RDC problem for its potential. This is done by applying the convolution parse tree kernel to incorporate various entity-related features into syntactic structure of relation examples. Different from Zhang et al (2006), we utilize a single expanded convolution parse tree kernel instead of a composite kernel. Different from Zhou et al. (2007), where PT is extended, we apply several linguistic rules to further prune out the noisy information from PT in better representing relation instances. One of our motivations is to capture both syntactic and semantic information in a single parse tree for further graceful refinement with the other to avoid the difficulty with tuning parameters in composite kernels. Empirical evaluation on the ACE2004 corpus shows that our method works well and achieves promising performance compared with other kernel-based systems.

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 Parse Tree Kernel

In this section, we first describe the expanded convolution parse tree kernel and demonstrate how entity-related features can be incorporated into the parse tree. Then, several linguistic rules are proposed to apply on the parse tree to further prune out the noisy information.

We adopt the same convolution tree kernel used by Collins and Duffy (2001), Moschitti (2004), Zhang et al. (2006) and Zhou et al. (2007). This convolution tree kernel counts the number of sub-trees that have similar productions on every node between two parse trees. T_1 and T_2 :

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

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

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

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

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

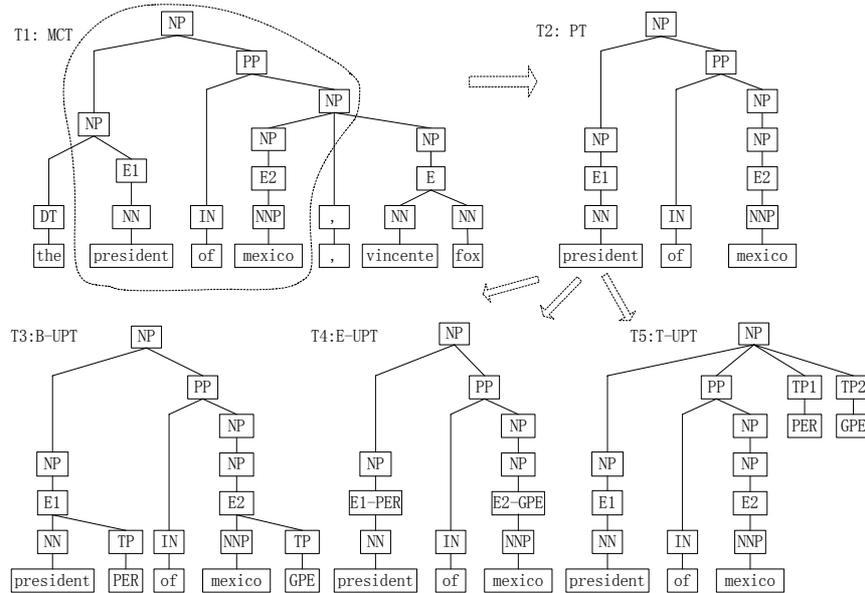


Figure 1: Different representations of a relation instance

From the 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 tree setups to extract the portion of a parse tree for relation extraction. Their experiments show that PT

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.

(Path-enclosed Tree) achieves best performance among those cases. In this paper, we begin with PT and then explore incorporating entity features at different locations. Our motivation is to evaluate whether and how entity information will be useful for relation extraction and in what way we can incorporate the entity information (especially the location where we attach) in the parse tree in order to achieve better performance. As depicted in Figure 1, the fragment “the president of mexico, vincente fox” is excerpted from the ACE2004 corpus, where a relation “EMP-ORG.employ-Executive” holds between the first entity “president”(PER) and the second entity “mexico” (GPE). The corresponding tree setups are listed as follows:

- (1) Minimum Complete Tree (MCT, T1 Fig. 1): the complete sub-tree rooted by the nearest common ancestor of the two entities. We include this case here for completeness.
- (2) Path-enclosed Tree (PT, T2 in Fig.1): Similar to the PT in Zhang et al. (2006), we also applies PT as the baseline.
- (3) Bottom-attached UPT (B-UPT, T3 in Fig.1): the entity type information is attached to the bottom of the entity node, i.e., two more nodes with the “TP” tags are added under the first and the second entity nodes respectively.
- (4) Entity-attached CPT (E-UPT, T4 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.
- (5) Top-attached CPT (T-UPT, T5 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.

The major problems with various UPTs are that they may still contain much noisy information and that some important contextual information may be missing. Based on the empirical observation that different relation types may have different kinds of contextual information and syntactic structures, we have attempted three kinds of compression operations to prune the UPTs in terms of linguistic knowledge:

- (1) DEL_ENT2_PRE: Removing the constituents (except the head) of the 2nd entity. Normally, the head of a noun phrase for an entity plays a key role in relation extraction. For example, in the noun phrase “the families of *seven other former* hostages”, “families”(PER) and “hostages”(PER) are heads for two entities respectively. Together with preposition “of”, they uniquely determine the relation type “PER-SOC.Family”. Other constituents (except the head) of the 2nd entity (e.g. the fragment “*seven other former*”) can be removed from the parse tree if the 2nd entity is a noun phrase and the two entities are not embedded in each other.
- (2) CMP_NP_CC_NP: Compressing noun phrase coordination conjunction. This is to compress coordinated conjuncts into a single one. Noun phrase coordination conjunctions occur frequently in the ACE RDC 2004 corpus, e.g. “governors from *connecticut, south dakota, and montana*”, where a relation “EMP-ORG.employ-Executive” exists between “governors”(PER) and “montana”(GPE). Figure 2 illustrates the parse trees before and after compression.
- (3) CMP_SINGLE_INOUT: Compressing single in-and-out nodes. Although Zhang et al. (2006) shows that noun phrase nodes with single in-and-out arcs does not help in relation extraction, we re-evaluate its impact in our experimental settings and further compress “X-->Y-->Z” into “X-->Z”.

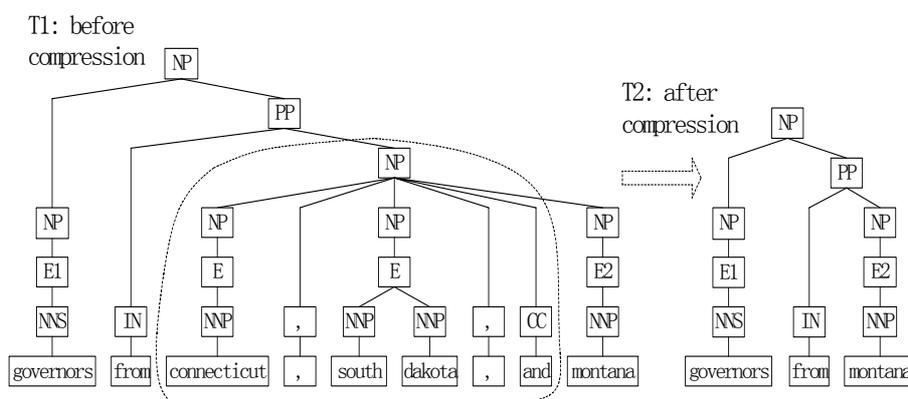


Figure 2. Compression of noun phrase coordination conjunctions

3 Experiments

We have implemented the proposed tree kernel method and systematically on the ACE RDC 2004 corpus.

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 the Charniak parser (Charniak, 2001). Then, we iterate over all pairs of entity mentions occurring in the same sentence to generate potential relation instances.

For classifiers, we choose binary-class SVM (Vapnik 1998) 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 classifiers to multi-category classifiers. The final decision of a relation instance in the multi-category 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 also set to 2.4 and 0.4 respectively.

3.2 Experimental Results

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

Different instance representations

According to the above discussion, we select PT with entity order information as our baseline in attempting to discover whether and how entity information will be effective to relation extraction. In order to reduce the training time, we only add major type information into the

parse tree. Table 1 compares the performance of seven major types for five different setups in the ACE2004 corpus using the expanded convolution tree kernel. It shows that:

- Using the tree kernel with PT achieves the performance of 68.3%/51.3%/58.6 in precision/recall/F-measure. This indicates the effectiveness of the convolution parse tree kernel and the PT in relation extraction.
- Compared with MCT and PT, UPTs significantly improves the F-measure due to the increase both in precision and recall. This indicates that the UPTs incorporated with entity type information produce significant performance improvement for relation extraction.
- Among the three UPTs, T-UPT achieves better performance (1.5/1.9 in F-measure) than the other two. This may be due to the introduction of the decay factor λ (set to 0.4 here) in making the kernel less dependent on the tree size, which decreases the effect of entity-related information when attached at lower nodes. Therefore in the following, we will apply the T-UPT by default.

Tree Setup	P	R	F
MCT	68.4	43.8	53.4
PT	68.3	51.3	58.6
B-UPT	75.3	59.9	66.7
E-UPT	76.3	59.8	67.1
T-UPT	76.8	62.1	68.6

Table 1: Comparison of tree setups on the major types of the ACE RDC 2004 corpus

Different entity features

In addition to the entity type, there are many other entity features on an entity, such as subtype, mention level and entity class. One may wonder whether they have different contributions to relation extraction. We will answer this question in the following.

Table 2 evaluates the contributions of different kinds of entity-related information, such as entity type, subtype, mention level and entity class, on the seven major types of the ACE RDC 2004 corpus using the T-UPT by adding them one by one in the decreasing order of their potential importance. It indicates that our system achieves the best performance of 80.6%/65.1%/72.0 in precision/recall/F-measure respectively. It also shows that:

- Entity type information is very effective and significantly increases the F-measure by 10 units. This may be due to that entity type information imposes strong constraints on relation types.
- Entity subtype and mention level information moderately improves the F-measure by 1.2 and 1.6 units respectively. This further shows that gracefully defined entity type and subtype information contribute much to performance improvement.
- It is a bit surprising to observe that other four kinds of information, including entity class, GPE role, head word and LDC mention type, decrease the F-measure by 0.4/0.3/1.0/1.0 units respectively. This suggests that such information may be too detailed (for head word) or non-discriminative (for other three) to make positive effect.
- Finally, we also evaluate the contribution of the predicate verb (in basic form) nearest to the second entity. It shows that such predicate-related information slightly improves the F-measure by 0.6 units, largely due to the increase in recall. This suggests that moving verbs from the bottom to the top of the parse tree may help relation extraction.

Entity info.	P	R	F
+major type*	76.8	62.1	68.6
+subtype*	78.6	62.7	69.8
+mention level*	80.4	64.2	71.4
+entity class ⁽⁻⁾	80.8	63.4	71.0
+GPE role ⁽⁻⁾	80.4	63.8	71.1
+head word ⁽⁻⁾	81.5	61.9	70.4
+LDC type ⁽⁻⁾	79.9	62.9	70.4
+predicate base	80.6	65.1	72.0

Table 2: Contribution of different entity features over seven major types in the ACE2004 corpus using the above T-UPT setup.

(Note: The aster sign 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.)

Different linguistic rules

Table 3 further compares the contributions of different linguistic rules on the seven major types of the ACE RDC 2004 corpus using the T-UPT by adding them one by one. It shows that our system achieves the best performance of 79.0%/68.0%/73.1 in precision/recall/F-measure respectively. This suggests that removing noisy information from the parse tree does help in relation extraction. It also indicates that:

Linguistic Rules	P	R	F
UPT (baseline)	80.6	65.1	72.0
+DEL_ENT2_PRE	79.2	66.7	72.4
+CMP_SINGLE_INOUT	79.2	67.3	72.8
+CMP_NP_CC_NP	79.0	68.0	73.1

Table 3. Contributions of different rules on the major types of the ACE RDC 2004 corpus using the T-UPT setup.

- Removing the constituents (except the head) of the 2nd entity improves the F-measure by 0.4, due to the increase in recall, even though it decreases the precision by 1.4%.
- Compressing single in-and-out nodes improves the F-measure by 0.4 units, due to the increase in recall. This means that careful handling of the nodes with single in-and-out arcs is helpful in relation extraction.
- Compressing noun phrase coordination conjunctions can simplify the syntactic structure and slightly improve the performance by 0.4 in F-measure.

Comparison with related work

Table 4 compares our system with recently best-reported work on the ACE2004 corpus. It shows that our system achieves promising performance. It is not surprising that the composite kernel (Zhou et al, 2007) performs better than ours, since it integrates much more flat features. Nevertheless, our kernel achieves nearly the same performance as context-sensitive tree kernel (Zhou et al, 2007), this means that both extending necessary information and removing noisy information can contribute much to relation extraction. Compared with the composite kernel (Zhang et al, 2006), our system further prunes the parse

tree and incorporates entity-related features into the convolution parse tree kernel. It indicates that our system achieves higher precision, lower recall and slightly better F-measure than their method. Compared with feature-based systems (Zhou et al, 2005 and Zhao et al, 2005) that incorporate many lexical, syntactic and semantic features, our system improves the F-measure by 2.1 and 2.8 units over major 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.

Systems	RDC on Major Types		
	P	R	F
Ours: expanded tree kernel	79.0	68.0	73.1
Zhou et al (2007): composite kernel	82.2	70.2	75.8
Zhou et al (2007): context-sensitive tree kernel	81.1	66.7	73.2
Zhang et al (2006): composite kernel	76.1	68.4	72.1
Zhou et al (2005): feature-based	82.8	62.1	71.0
Zhao et al (2005): feature-based	69.2	70.5	70.3

Table 4: Comparison of our system with best-reported ones on the ACE RDC 2004 corpus

4 Conclusion and Future Work

In this paper, we have modified a previous convolution parse tree kernel by incorporating entity-related features and pruning out noisy information for relation extraction using Support Vector Machines. Empirical evaluation on the ACE2004 corpus shows that the expanded convolution parse tree kernel works well and achieves promising performance, compared with other kernel-based systems. Our contributions lie in two points: first, the modified convolution parse tree (incorporated with entity-related features) significantly improves performance. Our evaluation shows that the higher we put entity feature nodes in the parse tree, the better performance we can get. Second, removing noisy information from the parse tree (according to linguistic rules) does contribute to relation extraction, and further detailed work may be done to reduce more noisy information inherent in the parse tree.

In the future work, we will attempt to construct a dynamic relation tree in order to reflect both the syntactic structure and semantic information more accurately in terms of lexical dependency theory and discourse theory. We will also utilize semantic resources (such as WordNet) to explore semantic similarity between terminal words (e.g. noun for entity and verb for predicate respectively) in the parse tree.

Acknowledgements

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

References

- Bunescu, R. C. and Mooney, R. J. 2005. A Shortest Path Dependency Kernel for Relation Extraction. In Proceedings of HLT/EMNLP-2005, pp. 724-731.
- Charniak, E., 2001. Intermediate-head Parsing for Language Models. In Proceedings of ACL-2001, pp. 116-123.
- Collins, M. and Duffy, N. 2001. Convolution Kernels for Natural Language. In Proceedings of NIPS-14, pp. 625-632.
- Culotta, A. and Sorensen, J. 2004. Dependency tree kernels for relation extraction. In 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. In 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. In Proceedings of ACL-2004(posters), pp. 178-181.
- Moschitti, A. 2004. A Study on Convolution Kernels for Shallow Semantic Parsing. In Proceedings of ACL-2004, pp. 335-342.
- Ting, W., Yaoyong, L., Kalina, B., Hamish, C., and Ji, W. 2006. Automatic Extraction of Hierarchical Relations from Text. In 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., Aone, C. and Richardella, A. 2003. Kernel Methods for Relation Extraction. *Journal of Machine Learning Research*, 2(2003), 1083-1106.
- Zhang, M., Zhang, J., Su, J. and Zhou, G. D. 2006. A Composite Kernel to Extract Relations between Entities with both Flat and Structured Features. In Proceedings of ACL-2006, pp. 825-832.
- Zhao, S.B. and Grisman, R. 2005. Extracting relations with integrated information using kernel methods. In Proceedings of ACL-2005, pp. 419-426.
- Zhou, G. D., Su, J., Zhang, J. and Zhang, M. 2005. Exploring various knowledge in relation extraction. In Proceedings of ACL-2005, pp. 427-434.
- Zhou, G. D., Zhang, M., Ji, D. H. and Zhu, Q. M. 2007. Tree Kernel-based Relation Extraction with Context-Sensitive Structured Parse Tree Information. In Proceedings of EMNLP-2007, pp. 728-736.