COSI 137 Information Extraction - Final Project: Event Relation Extraction in Syntax Constructions using Tree Kernel

Chuan Wang
Computer Science Department,
Brandeis University

Abstract

In this project, I mainly focus on the event pairs in verb-clause structures within TimeML corpus and try to apply tree kernel method to the event relation extraction task. With comparing the results given by tree kernel only, combined kernel and traditional manual feature engineering method, we conclude that tree kernel is still not comparable to the traditional feature engneering way, which proves that feature engineering with linguistic intuition is still the more practical way and give us the more better performance.

1 Introduction

Event relation extraction task is to find the temporal relation between events in natural language processing. In order to fully understand the text for machine, we not only depend on entity recognition, syntactic parsing and semantic role labeling, but also have to know temporal structures within text. TimeML[8], which contains parts of the temporal structure of the 186 document Timebank corpus, aims to provide researchers the structured data to train models for extracting events and temporal structure. By utilizing machine learning with some feature engineering techniques, reseachers have achieved performance of precision and recall around 70% and 80% for the event detection task[1]. However, event relation i.e. temporal ordering could be more challenging because of the data sparsity and annotation inconsistency[6].

[2] suggests that temporal relation guided by linguistic knowledge should give better performance and they focus on the event pairs within specific types of patterns on a syntactic tree, which is called *syntactic constructions*. They examine temporal relations expressed through the verb-clause construction (see Figure 1) in TimeML corpora and re-annotate them with label *BEFORE,AFTER,OVERLAP*. Then they train a temporal relation identification model with manually selected two set of features. One set gives the linguistic description of each isolated events. The other set characterize the relation labelings of two events.

My work mainly follows the track of [2]. However, rather than only using the manually constructed feature set, which requires a lot of lingustic intuition and feature selection, we examine the tree kernel method to estimate the similarity between two syntatic subtrees which dominate two events. Another advantage of tree kernel method is that they can search a much larger feature space than manual feature engineering can construct. And by recursing each subtrees or subset trees within syntax tree, we are able to discover implicit features which cannot be provided by manual feature engineering. There have been a lot of work on using tree kernel method to address relation extraction task. In [4], they construct dependency tree kernels and add different features into the dependency tree nodes, and then estimate the similarity between two subtrees by using feature matching function. Another type of tree kernel which has been successfully utilized in the predicate argument classification task is building tree kernels on syntax tree and the tree kernel function calculates the similarity by considering the number of common fragments btween two substructures recursively[7]. Since predicate argument is also a kind of relation between the verb and its argument, in this project I mainly use the syntax tree kernel to address the event relation identification task. Also, I have also combined the tree kernel with the common SVM kernel with low leverl manual features, which

2 Syntax Parse Tree Kernel

The tree kernel I examined is mainly from work of [7]. First they represent trees as substructures (fragments). The substructures are then mapped into vector spaces. The similarity between two trees are calculated by counting the number of their common fragments. There are two kind of fragment types: the subtrees (STs) and the

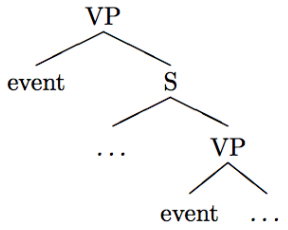

Figure 1: The verb-clause syntactic construction

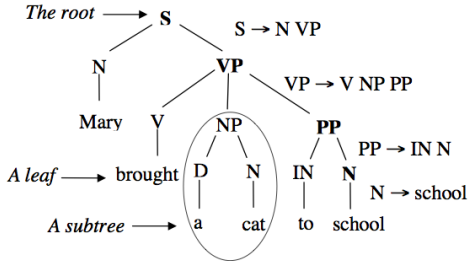

Figure 2: A syntactic parse tree

subset trees (SSTs).

2.1 Subtrees and Subset Trees

The tree kernel is built on the syntactic tree which is generated according to the grammar production rule. For example, the syntax tree for the sentence `"Mary brought a cat to school"` is shown in Figure 2.

The subtree (ST) is any node of a tree along with all its descendants as illustrated in circle of Figure 2. A subset tree (SST) is a more general structure in which a node don't have its all descendants. Its leaves can also be non-terminal symbols. The differences bwteen subtree and subset tree is illustrated in Figure 3 and Figure 4.

The subtrees (STs) and subset trees (SSTs) can be regarded as two different ways of representing the features extracted from syntax tree. But rather than manually encoding them, we can automatically extract them and measure the different information which is quantified by the number of different substructures.

2.2 The Tree Kernel Functions

In this section we give the description of tree kernel functions defined in [7]. Given the set of fragments (substructures) $\{f_1, f_2, \ldots\} = F$, the defination of tree kernel is given as,

$$K(T_1, T_2) = \sum_{n_1 \in N_{T_1}} \sum_{n_2 \in N_{T_2}} \phi(n_1, n_2) \qquad (1)$$

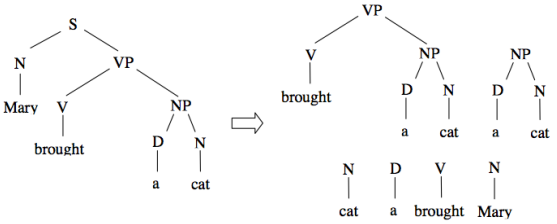

Figure 3: A syntactic parse tree with its subtrees (STs)

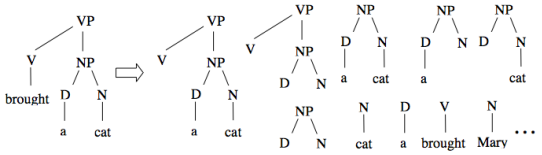

Figure 4: A tree with some of its subset trees (SSTs)

where N_{T_1} and N_{T_2} are the node sets of tree T_1 and T_2. And $\phi(n_1, n_2) = \sum_{i=1}^{|F|} I_i(n_1) I_i(n_2)$, where $I_i(n)$ is the indicator function which is equal to 1 if the target f_i is rooted at node n and 0 otherwise.. The right hand side actually counts the number of common fragments rooted in n_1 and n_2. And, [3] have shown that $\phi(n_1, n_2)$ can be computed in polynomial time using the following recursive definition:

- if the productions at n_1 and n_2 are different then $\phi(n_1, n_2) = 0$
- if the productions at n_1 and n_2 are the same, and n_1 and n_2 have only leaf children (i.e. they are pre-terminals symbols) then $\phi(n_1, n_2) = 1$;
- if the productions at n_1 and n_2 are the same, and n_1 and n_2 are not pre-terminals then

$$\phi(n_1, n_2) = \Pi_{j=1}^{nc(n_1)} (1 + \phi(c_{n_1}^j, c_{n_2}^j)) \qquad (2)$$

where $nc(n_1)$ is the number of the children of n_1 and c_n^j is the j-th child of the node n.

3 Experiments

I mainly use `SVM-light`[7] tree kernel package to do this experiment. `SVM-light-tk` is built on the original `SVM-light` packaage[5]. So it supports the combination of different kernels. For example, we can combine the tree kernel with the SVM polynomial kernel with hand selected features. Later I will explain the combination model for comparison. Note that the tree kernel described is originally used in the predicate argument

classification task. Here we treat the event pairs in the verb-clause as the predicate-argument pairs, since both of there are in the syntax structures.

3.1 Experiment Settings

I mainly use the 132 newswire documents in Wall Street Journal section of the TimeBank with gold-standard labeled events and relations between them. I also pick out all the compatible gold-standard syntactic trees from the TreeBank. To select the event pairs within verb-clause structure, I first find the path along this two events and then find out the part-of-speech tag sequence along the path. Then I use some simple regular expressions to select the target event pairs.

Since the some of the labels are very sparse and may share the same meaning, I merge them for simplicity. For Label BEFORE and AFTER, because these two are symmetric , I just treat them as one label. For label SIMULTANEOUS, DURING, IS_INCLUDED and INCLUDES, I merge them into one label SIMUL. And Table 1 shows the data distribution with the labels.

Because SVM-light-tk package can only handle binary classification, I build an individual ONE-va-ALL(OVA) classifier C_i for each label i. As a final decision of the multiclassfier, I select the label j associated with the maximum value among all the scores given by C_i,i.e.$j = \arg max_{i \in Y} score(C_i)$, where Y is the set of labels. I

Class	Train	Test
AFTER/BEFORE	99	46
SIMUL	65	50
MODAL	225	91
EVIDENTIAL	300	141
Total	**689**	**424**

Table 1: Data distribution of the selected event pairs

mainly do three experiment: 1) use the tree kernel only with extracting the minimum subtree which dominates the two events. 2) combine the tree kernel with SVM poly nominal kernel trained with manual feature set 3) use the polynominal kernel only with manual feature set.

The manual feature set includes:

- **Tense, aspect, polarity, class of each event**: these have been annotated in the TimeML corpus
- **word, pos, stem of each event**
- **compl-word**:The text of the complementizer for the clause, e.g. to, that or because.
- **compl-type**: The type of the complementizer, determined by a simple set of rules.

3.2 Results

Class	Precision	Recall	F1
AFTER/BEFORE	0.2745	0.3043	0.288
SIMUL	0.3888	0.28	0.33
MODAL	0.7954	0.7692	0.78
EVIDENTIAL	0.8039	0.8723	0.84

Table 2: Tree Kernel only classfication results

Class	Precision	Recall	F1
AFTER/BEFORE	0.4464	0.5435	0.49
SIMUL	0.5	0.24	0.32
MODAL	0.9277	0.8462	0.885
EVIDENTIAL	0.83	0.97	0.894

Table 3: Tree Kernel + manual features classfication results

Class	Precision	Recall	F1
AFTER/BEFORE	0.4375	0.6086	0.509
SIMUL	0.411	0.14	0.2089
MODAL	0.939	0.8461	0.89
EVIDENTIAL	0.83	0.97	0.8954

Table 4: Manual features only classfication results

From Table 2, we can see that the performance greatly suffers from the sparsity of the data. Tree kernel performance on labels MODAL and EVIDENTIAL is far better that the sparse label AFTER/BEFORE and SIMUL. Compared to Table 2, the combined kernel performs better on most of the labels with only minor drop on the SIMUL label. Also look into the performance given by only using the hand selected features, it achieves almost the same F1 measure on most of the labels except for the most sparse label SIMUL.

It seems that tree kernel is not so impressive as what the intuition says. The linguistic intuition is still far more helpful than the features which are generated by applying structure comparison. Even some researchers have proved that the tree kernel can be treated as an extra feature which will always improve the performance of models with manual feature sets[7], it is at least not proved in this experiment. It is also possible that only comparing and calculating the similarity of structure information is not enough. We should extend the feature space not only about the syntax structure information but also some semantic or word knowledge, which is really hard to encode without linguistic intuition.

3.3 Future Work

It may be helpful to train some other methods of choosing the trees that are related to the event pairs. Since in this project I only tried the minimum dominate tree. It is very insteresting what the performance will be when we apply more tree structure like tree path or tree nodes with only one child which is an event. Also, it may be necessary for us to compare the dependency tree kernel which import features into nodes with the combined kernel discussed in this project.

References

[1] BETHARD, S., AND MARTIN, J. H. Identification of event mentions and their semantic class. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing* (Stroudsburg, PA, USA, 2006), EMNLP '06, Association for Computational Linguistics, pp. 146–154.

[2] BETHARD, S., MARTIN, J. H., AND KLINGENSTEIN, S. Finding temporal structure in text: Machine learning of syntactic temporal relations. *International Journal of Semantic Computing 1*, 04 (2007), 441–457.

[3] COLLINS, M., AND DUFFY, N. Convolution kernels for natural language. In *Proceedings of NIPS* (2001), vol. 14, pp. 625–632.

[4] CULOTTA, A., AND SORENSEN, J. Dependency tree kernels for relation extraction. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics* (2004), Association for Computational Linguistics, p. 423.

[5] JOACHIMS, T. Making large scale svm learning practical.

[6] MANI, I., VERHAGEN, M., WELLNER, B., LEE, C. M., AND PUSTEJOVSKY, J. Machine learning of temporal relations. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics* (2006), Association for Computational Linguistics, pp. 753–760.

[7] MOSCHITTI, A. Making tree kernels practical for natural language learning. In *Proceedings of EACL* (2006), vol. 6, pp. 113–120.

[8] PUSTEJOVSKY, J., CASTANO, J., INGRIA, R., SAURI, R., GAIZAUSKAS, R., SETZER, A., KATZ, G., AND RADEV, D. Timeml: Robust specification of event and temporal expressions in text. *New directions in question answering 3* (2003), 28–34.