Composite Kernels For Relation Extraction

Frank Reichartz
Fraunhofer IAIS
St. Augustin, Germany

Hannes Korte
Fraunhofer IAIS
St. Augustin, Germany

Gerhard Paass
Fraunhofer IAIS
St. Augustin, Germany

{frank.reichartz,hannes.korte,gerhard.paass}@iais.fraunhofer.de

Abstract

The automatic extraction of relations between entities expressed in natural language text is an important problem for IR and text understanding. In this paper we show how different kernels for parse trees can be combined to improve the relation extraction quality. On a public benchmark dataset the combination of a kernel for phrase grammar parse trees and for dependency parse trees outperforms all known tree kernel approaches alone suggesting that both types of trees contain complementary information for relation extraction.

1 Introduction

The same semantic relation between entities in natural text can be expressed in many ways, e.g. “Obama was educated at Harvard”, “Obama is a graduate of Harvard Law School”, or, “Obama went to Harvard College”. Relation extraction aims at identifying such semantic relations in an automatic fashion.

As a preprocessing step named entity taggers detect persons, locations, schools, etc. mentioned in the text. These techniques have reached a sufficient performance level on many datasets (Tjong et al., 2003). In the next step relations between recognized entities, e.g. person-educated-in-school(Obama,Harvard) are identified.

Parse trees provide extensive information on syntactic structure. While feature-based methods may compare only a limited number of structural details, kernel-based methods may explore an often exponential number of characteristics of trees without explicitly representing the features. Zelenko et al. (2003) and Culotta and Sorensen (2004) proposed kernels for dependency trees (DTs) inspired by string kernels. Zhang et al. (2006) suggested a kernel for phrase grammar parse trees. Bunescu and Mooney (2005) investigated a kernel that computes similarities between nodes on the shortest path of a DT connecting the entities. Reichartz et al. (2009) presented DT kernels comparing substructures in a more sophisticated way.

Up to now no studies exist on how kernels for different types of parse trees may support each other. To tackle this we present a study on how those kernels for relation extractions can be combined. We implement four state-of-the-art kernels. Subsequently we combine pairs of kernels linearly or by polynomial expansion. On a public benchmark dataset we show that the combined phrase grammar parse tree kernel and dependency parse tree kernel outperforms all others by 5.7% F-Measure reaching an F-Measure of 71.2%. This result shows that both types of parse trees contain relevant information for relation extraction.

The remainder of the paper is organized as follows. In the next section we describe the investigated tree kernels. Subsequently we present the method to combine two kernels. The fourth section details the experiments on a public benchmark dataset. We close with a summary and conclusions.

2 Kernels for Relation Extraction

Relation extraction aims at learning a relation from a number of positive and negative instances in natural language sentences. As a classifier we use Support Vector Machines (SVMs) (Joachims, 1999) which can compare complex structures, e.g. trees, by kernels. Given the kernel function, the SVM tries to find a hyperplane that separates positive from negative examples of the relation. This type of max-margin separator has been shown both empirically and theoretically to provide good generalization performance on new examples.

Proceedings of the ACL-IJCNLP 2009 Conference Short Papers, pages 365–368,
Suntec, Singapore, 4 August 2009. ©2009 ACL and AFNLP

2.1 Parse Trees

A sentence can be processed by a parser to generate a parse tree, which can be further categorized in phrase grammar parse trees (PTs) and dependency parse trees (DTs). For DTs there is a bijective mapping between the words in a sentence and the nodes in the tree. DTs have a natural ordering of the children of the nodes induced by the position of the corresponding words in the sentence. In contrast PTs introduce new intermediate nodes to better express the syntactical structures of a sentence in terms of phrases.

2.2 Path-enclosed PT Kernel

The *Path-enclosed PT Tree Kernel* (Zhang et al., 2008) operates on PTs. It is based on the *Convolution Tree Kernel* of Collins and Duffy (2001). The *Path-enclosed Tree* is the parse tree pruned to the nodes that are connected to leaves (words) that belong to the path connecting both relation entities. The leaves (and connected inner nodes) in front of the first relation entity node and behind the second one are simply removed. In addition, for the entities there are new artificial nodes labeled with the relation argument index, and the entity type. Let $K_{\mathrm{CD}}(T_1, T_2)$ be the *Convolution Tree Kernel* (Collins and Duffy, 2001) of two trees T_1, T_2, then the *Path-enclosed PT Kernel* (ZhangPT) is defined as

$$K_{\mathrm{ZhangPT}}(X, Y) = K_{\mathrm{CD}}(X^*, Y^*)$$

where X^* and Y^* are the subtrees of the original tree pruned to the nodes enclosed by the path connecting the two entities in the phrase grammer parse trees as described by Zhang et al. (2008).

2.3 Dependency Tree Kernel

The *Dependency Tree Kernel* (DTK) of Culotta and Sorensen(2004) is based on the work of Zelenko et al. (2003). It employs a *node kernel* $\Delta(u, v)$ measuring the similarity of two tree nodes u, v and its substructures. Nodes may be described by different features like POS-tags, chunk tags, etc.. If the corresponding word describes an entity, the entity type and the mention is provided. To compare relations in two instance sentences X, Y Culotta and Sorensen (2004) proposes to compare the subtrees induced by the relation arguments x_1, x_2 and y_1, y_2, i.e. computing the node kernel between the two lowest common ancestors (lca) in the dependecy tree of the relation argument nodes

$$K_{\mathrm{DTK}}(X, Y) = \Delta(lca(x_1, x_2), lca(y_1, y_2))$$

The node kernel $\Delta(u, v)$ is definend over two nodes u and v as the sum of the node similarity and their children similarity. The *children similarity function* $C(s, t)$ uses a modified version of the *String Subsequence Kernel* of Shawe-Taylor and Christianini (2004) to compute recursively the sum of node kernel values of subsequences of node sequences s and t. The function $C(s, t)$ sums up the similarities of all subsequences in which every node matches its corresponding node.

2.4 All-Pairs Dependency Tree Kernel

The All-Pairs Dependency Tree Kernel (All-Pairs-DTK) (Reichartz et al., 2009) sums up the node kernels of all possible combinations of nodes contained in the two subtrees implied by the relation argument nodes as

$$K_{\text{All-Pairs}}(X, Y) = \sum_{u \in V_x} \sum_{v \in V_y} \Delta(u, v)$$

where V_x and V_y are sets containing the nodes of the complete subtrees rooted at the respective lowest common ancestors. The consideration of all possible pairs of nodes and their similarity ensure that relevant information in the subtrees is utilized.

2.5 Dependency Path Tree Kernel

The Dependency Path Tree Kernel (Path-DTK) (Reichartz et al., 2009) not only measures the similarity of the root nodes and its descendents (Culotta and Sorensen, 2004) or the similarities of nodes on the path (Bunescu and Mooney, 2005). It considers the similarities of all nodes (and substructures) using the node kernel Δ on the path connecting the two relation argument entity nodes. To this end the pairwise comparison is performed using the ideas of the subsequence kernel of Shawe-Taylor and Cristianini (2004), therefore relaxing the "same length" restriction of (Bunescu and Mooney, 2005). The Path-DTK effectively compares the nodes from paths with different lengths while maintaining the ordering information and considering the similarities of substructures.

The parameter q is the upper bound on the node distance whereas the parameter μ, $0 < \mu \leq 1$, is a factor that penalizes gaps. The Path-DTK is

	5-times 5-fold Cross-Validation on Training Set						Test Set					
Kernel	At	Part	Role	Prec	Rec	F	At	Part	Role	Prec	Rec	F
DTK	54.9	52.8	72.3	71.7	53.7	61.4 (0.32)	50.3	43.4	68.5	79.5	44.0	56.7
All-Pairs-DTK	59.1	53.6	73.0	73.1	57.8	64.5 (0.26)	54.3	53.9	71.8	80.2	49.6	61.3
Path-DTK	64.8	62.9	77.2	80.2	61.2	69.4 (0.09)	54.9	55.6	73.5	76.7	52.8	62.5
ZhangPT	66.8	69.1	77.7	80.6	65.0	71.9 (0.21)	62.9	64.2	72.2	82.0	54.5	65.5
ZhangPT + Path-DTK	**70.1**	**76.6**	**80.8**	**84.6**	**68.2**	**75.5** (0.20)	**66.3**	**71.3**	**77.7**	**85.7**	**60.9**	**71.2**

Table 1: F-values for 3 selected relations and micro-averaged precision, recall and F-score (with standard error) for all 5 relations on the training (CV) and test set in percent.

defined as

$$K_{\text{Path-DTK}}(X,Y) = \sum_{\substack{\mathbf{i}\in I_{|x|},\mathbf{j}\in I_{|y|},\\ |\mathbf{i}|=|\mathbf{j}|,\, d(\mathbf{i}),d(\mathbf{j})\le q}} \mu^{d(\mathbf{i})+d(\mathbf{j})}\Delta'(x(\mathbf{i}),y(\mathbf{j}))$$

where x and y are the paths in the dependency tree between the relation arguments and $x(\mathbf{i})$ is the subsequence of the nodes indexed by **i**, analogously for **j**. I_k is the set of all possible index sequences with highest index k and $d(i) = max(\mathbf{i}) - min(\mathbf{i}) + 1$ is the covered distance. The function Δ' is the sum of the pairwise applications of the node kernel Δ.

3 Kernel composition

In this paper we use the following two approaches to combine two normalized[1] kernels K_1, K_2 (Schoelkopf and Smola, 2001). For a weighting factor α we have the composite kernel:

$$K_c(X,Y) = \alpha K_1(X,Y) + (1-\alpha)K_2(X,Y)$$

Furthermore it is possible to use polynomial expansion on the single kernels, i.e. $K^p(X,Y) = (K(X,Y)+1)^p$. Our experiments are performed with $\alpha = 0.5$ and the sum of linear kernels (L) or poly kernels (P) with $p = 2$.

4 Experiments

In this section we present the results of the experiments with kernel-based methods for relation extraction. Throughout this section we will compare the approaches considering their classification quality on the publicly available benchmark dataset ACE-2003 (Mitchell et al., 2003). It consists of news documents containing 176825 words splitted in a test and training set. Entities and the relations between them were manually annotated. The entities are marked by the types *named* (e.g. “Albert Einstein”) , *nominal* (e.g. “University”) and *pronominal* (e.g. “he”). There are 5 top level relation types *role*, *part*, *near*, *social* and *at*, which are further differentiated into 24 subtypes.

4.1 Experimental Setup

We implemented the tree-kernels for relation extraction in Java and used Joachim’s (1999) SVM^{light} with the JNI Kernel Extension using the implementation details from the original papers. For the generation of the parse trees we used the Stanford Parser (Klein and Manning, 2003). We restricted our experiments to relations between named entities, where NER approaches may be used to extract the arguments. Without any modification the kernels could also be applied to the all types setting as well. We conducted classification tests on the five top level relations of the dataset. For each relation we trained a separate SVM following the one vs. all scheme for multi-class classification. We also employed a standard grid-search on the training set with a 5-times repeated 5-fold cross validation to optimize the parameters of all kernels as well as the SVM-parameter C for the classification runs on the separate test set. We use the standard evaluation measures for classification accuracy: precision, recall and F-measure.

4.2 Results

Table 1 shows F-values for three selected relations and micro-averaged results for all 5 relations on the training and test set. In addition the F-scores for the three relations containing the most instances are provided. Kernel and SVM parameters are optimized solely on the training set. Note that the training set results were obtained on the left-out folds of cross-validation. The composite kernel ZhangPT + Path-DTK performs the best on the cross validations run as well as on the test-set. It outperforms all previously suggested solutions

[1] Kernel normalization: $K_n(X,Y) = \frac{K(X,Y)}{\sqrt{K(X,X)\cdot K(Y,Y)}}$

	DTK	All-Pairs-DTK	Path-DTK	ZhangPT
ZhangPT	63.5 (70.2) PP	67.9 (72.8) PP	**71.2** (**75.5**) LP	65.5 (71.9)
Path-DTK	62.7 (67.7) PP	62.9 (69.5) PL	62.5 (69.4)	
All-Pairs-DTK	60.0 (64.7) PP	61.3 (64.5)		
DTK	56.7 (61.4)			

Table 2: Micro-averaged F-values for the Single and Combined Kernels on the Test Set (outside parenthesis) and with 5-times repeated 5-fold CV on the Training Set (inside parenthesis). LP denotes the combination type linear and polynomial, analogously PP and PL.

by at least 5.7% F-Measure on the prespecified test-set and by 3.6% F-Measure on the cross validation. Table 2 shows the F-values of the different combinational kernels on the test set as well as on the cross validation on the training set. The ZhangPT + Path-DTK performs the best out of all possible combinations. The difference in F-values between ZhangPT + Path-DTK and ZhangPT is according to corrected resampled t-test (Bouckaert and Frank, 2004) significant at a level of 99.9%. These results show that the simultanous consideration of phrase grammar parse trees and dependency parse trees by the combination of the two kernels is meaningful for relation extraction.

5 Conclusion and Future Work

In this paper we presented a study on the combination of state of the art kernels to improve relation extraction quality. We were able to show that a combination of a kernel for phrase grammar parse trees and one for dependency parse trees outperforms all other published parse tree kernel approaches indicating that both kernels captures complementary information for relation extraction. A promising direction for future work is the usage of more sophisticated features aiming at capturing the semantics of words e.g. word sense disambiguation (Paaß and Reichartz, 2009). Other promising directions are the study on the applicability of the kernel to other languages and exploring combinations of more than two kernels.

6 Acknowledgement

The work presented here was funded by the German Federal Ministry of Economy and Technology (BMWi) under the THESEUS project.

References

Remco R. Bouckaert and Eibe Frank. 2004. Evaluating the replicability of significance tests for comparing learning algorithms. In *PAKDD '04*.

Razvan C. Bunescu and Raymond J. Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proc. HLT/EMNLP*, pages 724 – 731.

Michael Collins and Nigel Duffy. 2001. Convolution kernels for natural language. In *Proc. NIPS '01.*

Aron Culotta and Jeffrey Sorensen. 2004. Dependency tree kernels for relation extraction. In *ACL '04.*

Thorsten Joachims. 1999. Making large-scale SVM learning practical. In *Advances in Kernel Methods - Support Vector Learning*.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proc. ACL '03.*

Alexis Mitchell et al. 2003. ACE-2 Version 1.0; Corpus LDC2003T11. Linguistic Data Consortium.

Gerhard Paaß and Frank Reichartz. 2009. Exploiting semantic constraints for estimating supersenses with CRFs. In *Proc. SDM 2009.*

Frank Reichartz, Hannes Korte, and Gerhard Paass. 2009. Dependency tree kernels for relation extraction from natural language text. In *ECML '09.*

Bernhard Schoelkopf and Alexander J. Smola. 2001. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.*

John Shawe-Taylor and Nello Cristianini. 2004. *Kernel Methods for Pattern Analysis*. Cambridge University Press.

Erik F. Tjong, Kim Sang, and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *CoRR cs.CL/0306050:.*

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *J. Mach. Learn. Res.*, 3:1083–1106.

Min Zhang, Jie Zhang, and Jian Su. 2006. Exploring syntactic features for relation extraction using a convolution tree kernel. In *Proc. HLT/NAACL'06.*

Min Zhang, GuoDong Zhou, and Aiti Aw. 2008. Exploring syntactic structured features over parse trees for relation extraction using kernel methods. *Inf. Process. Manage.*, 44(2):687–701.