Relation Extraction for Monitoring Economic Networks

Martin Had, Felix Jungermann and Katharina Morik

Technical University of Dortmund
Department of Computer Science - Artificial Intelligence Group
Baroper Strasse 301, 44227 Dortmund, Germany

Abstract. *Relation extraction* from texts is a research topic since the message understanding conferences. Most investigations dealt with English texts. However, the heuristics found for these do not perform well when applied to a language with free word order, as is, e.g., German. In this paper, we present a German annotated corpus for *relation extraction*. We have implemented the state of the art methods of *relation extraction* using kernel methods and evaluate them on this corpus. The poor results led to a feature set which focusses on all words of the sentence and a tree kernel which includes words, in addition to the syntactic structure. The *relation extraction* is applied to monitoring a graph of economic company-directors network.

1 Introduction

Social networks have raised scientific attention, the goals ranging from enhancing recommender systems [4, 15, 5] to gaining scientific insights [6, 22]. Where the taggings, mailings, co-authorship, or citations in communities have well been investigated, the economic relationships between companies and their networking have less been studied.
Today's search engines are not prepared to answer questions like "show me all companies that have merged with Volkswagen". In order to get that information anyway, it would be necessary to do an extensive search and consider several sources. This is time consuming and tedious. This is why question answering approaches require automatic *relation extraction*.
Moreover, it is important to represent the extracted information in a compact and easily to access manner. Especially concerning *relation extraction*, the extracted entities and relations can be represented using an (un-)directed graph.
In this paper, we present an approach to monitoring economic information in the world wide web using a graph-based representation. We will show that it is possible to extract additional information using *relation extraction* techniques, which have not yet successfully been used on German texts, because German language features problems, which other languages – especially English – do not face. A comparison of our feature set and enhanced tree kernel with state of the art methods illustrates the importance of a balanced use of semantic and syntactic information. First, we describe the state of the art in *relation extraction*

Published in the Proceedings of the 14th International Conference on Applications of Natural Language to Information Systems (NLDB) 2009

using kernel methods, then we present our application, before we introduce or enhancement of the method and the experimental results.

2 Kernel Methods for Relation Extraction

The ACE RDC task [11] defines a relation as a valid combination of two entities that are mentioned in the same sentence and have a connection to each other. Relations may be symmetric or asymmetric. The schema of i relations in a sentence s is defined as follows:

Definition 1. Relation candidates *in a sentence:*

$$R_i(Sentence\ s) =< Type_m \in relationtypes,$$
$$(Argument_1 \in entities_s, Argument_2 \in entities_s) >$$

where $entities_s$ is the set of entities contained in the current sentence, and $relationtypes$ is the set of possible relations in the corpus.
Structured information of a sentence e.g. is the syntactic parse tree (an example can be seen in Figure 1), where each node follows a grammar production rule. By splitting up a tree in subtrees (see Figure 2) it is possible to calculate the

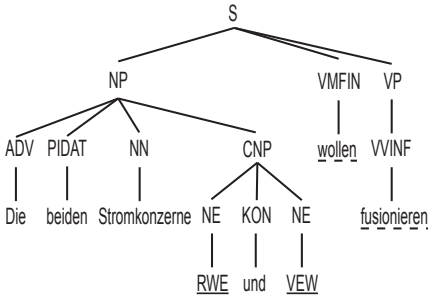

Fig. 1. A parse tree for a German sentence containing a merger-relation.

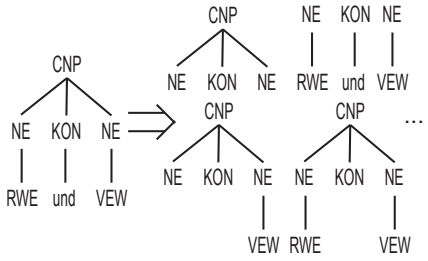

Fig. 2. Some subtrees of a tree

similarity of two trees by counting their common subtrees. The set of subtrees of a parse tree consists of every substructure that can be built by applying the grammatical rule set of the original tree.

2.1 Linear Kernels

First experiments on *relation extraction* have been done by just using feature-based methods. That made it necessary to manually create a large set of 'flat' features describing the relation and comparing the similarities of these feature-vectors in order to find the best discriminating classification function. The most

efficient way to compare feature-vectors is based on kernel functions which can be embedded in various machine-learning algorithms like support vector machines or clustering methods. A linear kernel on feature-vectors, x and z, is defined as their inner product:

Definition 2. *A* linear kernel*:*

$$K(x,z) = \sum_n \phi_n(x)\phi_n(z) \tag{1}$$

where $\phi_n(x)$ is the n-th feature of x.

2.2 Convolution Kernels

Converting syntactic structures into feature-vectors is tedious [21] [23]. This overhead is avoided when using a kernel function, which operates on any discrete structure [7]. Because of the formulation as a kernel, the calculation of the inner product requires the enumeration of substructures only implicitly.

Definition 3. *Suppose $x \in X$ is a composite structure and $\boldsymbol{x} = x_1, ..., x_p$ are its parts, where each $x_i \in X_i$ for $i = 1, ..., p$ and all X_i are countable sets. The relation $R(\boldsymbol{x}, x)$ is true, iff $x_1, ..., x_p$ are all parts of x. As a special case, X being the set of all p-degree ordered, rooted trees and $X_1 = ... = X_p = X$, the relation R can be used iteratively to define more complex structures in X.*
Given $x, z \in X$ and $\boldsymbol{x} = x_1, ..., x_p$, $\boldsymbol{z} = z_1, ..., z_p$ and a kernel K_i for X_i measuring the similarity $K_i(x_i, z_i)$, then the similarity $K(x, z)$ is defined as the following generalized convolution

$$K(x,z) = \sum_{\{\boldsymbol{x}|R(\boldsymbol{x},x)\}} \sum_{\{\boldsymbol{z}|R(\boldsymbol{z},z)\}} \prod_{i=1}^{p} K_i(x_i, z_i) \tag{2}$$

[7]p.5f

Convolution kernels characterize the similarity of parse trees by the similarity of their subtrees [3]. Within the kernel calculation, all subtrees of the trees are compared. They are (implicitly) represented as a vector:

$$\Phi(T) = (subtree_1(T), ..., subtree_m(T)) \tag{3}$$

where $subtree_i$ means the number of occurrences of the i-th subtree in T. The number of common subtrees is summed up. The worst case runtime is $O(|N_1| \times |N_2|)$, being N_t the set of nodes of a tree T_t.

Definition 4. *The* tree kernel *computes a scalar product:*

$$K(T_1, T_2) = < \mathbf{h}(T_1), \mathbf{h}(T_2) > \tag{4}$$

$$h_i(T_1) = \sum_{n_1 \in N_1} I_i(n_1) \tag{5}$$

where the indicator function I_i is defined for the nodes n_1 in N_1 of T_1 and n_2 in N_2 for T_2 as 1, iff the $i-th$ subtree is rooted in node n. Hence,

$$K(T_1, T_2) = \sum_{n_1 \in N_1} \sum_{n_2 \in N_2} \sum_i I_i(n_1) I_i(n_2) \tag{6}$$

$$= \sum_{n_1 \in N_1} \sum_{n_2 \in N_2} \Delta(n_1, n_2) \tag{7}$$

The calculation of Δ is done recursively by following three simple rules:

- If the grammar production rules of n_1 and n_2 are different: $\Delta(n_1, n_2) = 0$
- If the production rules in n_1 and n_2 are equal and n_1 and n_2 are pre-terminals (last node before a leaf): $\Delta(n_1, n_2) = \lambda$
- If the production rules in n_1 are n_2 equal and n_1 and n_2 are non pre-terminals:

$$\Delta(n_1, n_2) = \lambda \prod_{j=1}^{nc(n_1)} (1 + \Delta(ch(n_1, j), ch(n_2, j))) \tag{8}$$

$nc(n_1)$ = number of children of node n_1

$ch(n_1, i)$ = ith child of node n_1

λ = parameter to downweight the contribution of large tree fragments exponentially with their size.

[13] designed an algorithm for the above calculation that has linear runtime on average due to a clever preprocessing step. Nodes that don't need to be considered by the kernel are filtered out by sorting and comparing the production rules of both trees in advance ("Fast Tree Kernel" FTK).
[24] extended the FTK kernel to become context sensitive by looking back at the path above the ancestors of the root node of each subtree. The left side of the production rule is taking into account m-1 steps towards the root. The kernel calculation itself sums up the calculations for each set of production rules created for $1 \ldots m$. In the special case m=1 the kernel result is the same as with the non context-sensitive kernel.

$$K_C(T[1], T[2]) = \sum_{i=1}^{m} \sum_{n_1^i[1] \in N_i^1[1], n_1^i[2] \in N_i^1[2]} \Delta(n_1^i[1], n_1^i[2]) \tag{9}$$

- m the number of ancestor nodes to consider.
- $n_1^i[j]$ is a node of tree j with a production rule over i ancestors. $n_1[j]$ is the root node of the context free subtree the ancestor node of $n_k[j]$ is $n_{k+1}[j]$.
- $N_1^i[j]$ is the set of all nodes with their production rules over i ancestor.

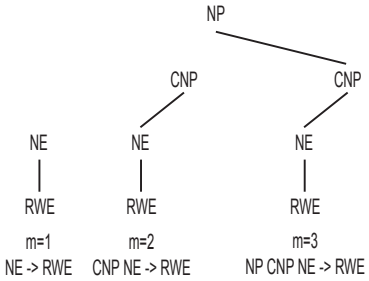

Fig. 3. Production rules of subtree root node (NE) for different m values

2.3 Composite Kernels

[20] showed that better results can be achieved with a combination of linear and tree kernels. [7] showed that the class of kernels is closed under product and addition. This implies that combining two kernels is possible and results in a new kernel which is called a *composite kernel*, defined as follows:

$$K(x, z) = K_1(x, z) \circ K_2(x, z) \tag{10}$$

In the case of *relation extraction* the composite kernel combines a linear and a convolution tree kernel. [20] propose a linear combination (11) and a polynomial expansion (12).

$$K(x, z) = \alpha \cdot \hat{K}_L(x, z) + (1 - \alpha) \cdot \hat{K}_T(x, z) \tag{11}$$

$$K(x, z) = \alpha \cdot \hat{K}_L^P(x, z) + (1 - \alpha) \cdot \hat{K}_T(x, z) \tag{12}$$

where x and z are *relation candidates* that consist of flat features and structured information, each kernel is given the right input for its kind. $K_L^P(x, z)$ is the polynomial expansion of $K_L(x, z)$ with degree $d = 2$ i.e. $K_L^P(x, z) = (K_L(x, z) + 1)^2$. By setting the α value the influence of each kernel type can be adjusted. Both kernels are normalized in kernel space before the combination:

$$K'(T_1, T_2) = \frac{K(T_1, T_2)}{\sqrt{K(T_1, T_2)K(T_1, T_2)}} \tag{13}$$

2.4 State-of-the-art composite kernels for Relation Extraction

In addition to just using the composite kernel on the full parse tree of a sentence, [20] examined several ways of pruning the parse tree in order to get differently shaped subtrees on which the treekernel performs as well or better as on the full tree. They showed that the shortest path-enclosed tree (SPT) which is the minimal subtree containing the two entities of a *relation candidate* performs best for the ACE 2003 RDC corpus.
But [24] showed that the ACE corpus contains relations for which the SPT is not

sufficient. These relations are indicated by their related verb. Figure 1 shows a relation of our corpus which is indicated by its related verb, too. But in contrast to the ACE corpus which just contains a few relations of this type, our corpus has many. The type of relation and the specialty of the German language are responsible for this fact.
The strict and binary decisions of the tree kernel are the main disadvantage of this method. [24] tried to overcome this problem by embedding syntactic features into the parse tree directly above the leaf-nodes. Moreover, syntactic structure is already covered by the tree kernel, adding it in terms of features does not help generalization. [19] generalized the production rules of the parse tree in order to achieve better performance. The strict decisions of a convolution tree kernel (remind Section 2.2) make the kernel returning 'unequal' confronted with two production rules "$NP \rightarrow Det\ Adj\ N$" and "$NP \rightarrow Det\ N$" although they might contain similar terminals ("$NP \rightarrow$ a red car" and "$NP \rightarrow$ a car"). To avoid such behavior they proposed inserting of optional nodes into production rules to generalize them. Additionally similar part of speech tags in the parse tree can be processed in an equal way – multiplied with a penalty term.
This is a step into the right direction. However, only syntactic variance is handled. Since words carry most of the semantic information, moving them into the tree kernel could well help to generalize in a more semantic way.

Related kernels for Relation Extraction There are several related approaches for *relation extraction* differing from the ones already presented. [17] presented a general kernel function for trees and its subtrees. [18] used a kernel function on shallow parse trees. Bunescu and Mooney used a kernel function on the shortest path between two entities in a dependency tree [1]. Additionally they used the context of entities for *relation extraction* [2].

3 Monitoring the merger event in an economic network

The enhancement of the state of the art in *relation extraction* which is described in Section 4 became necessary when we developed the economic network based on German sources. We did not want to manually build and update its database. Extracting relations from documents directly allows to automatically accomplish the data about companies and their board members with relationships between them. Hence, the network can be monitored and is always up-to-date.

3.1 Building-up the economic network

Building up the economic network starts with extracting companies and their board of directors. The extraction of the named entities "company" and "board member" is quite simple, because there exist several web archives of companies which are semi-structured. Hence, companies and their representatives can easily be extracted using simple regular expressions. The initial stock of data is stored in a SQL database. It consists of about 2.000 different big companies

from throughout the world. Basic information includes only address and industry but most entries provide a lot more details about members of the board of directorate, share ownership, shareholding and some key performance indicators. Many of these companies (here: 1,354) are connected to one other company at least, by sharing a member of the directorate. The best connected company even has 37 different outgoing directorate connections. These numbers support the assumption, that the graph built from these relations can reveal significant structures in the business world. From the SQL data base, a network $G = (V, E)$

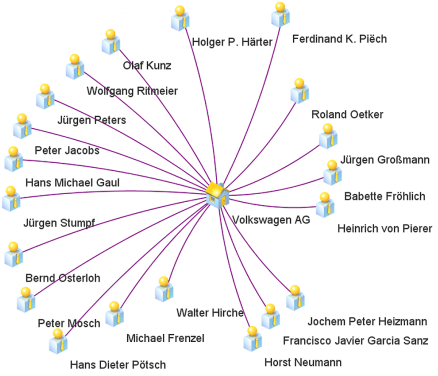

Fig. 4. Selecting Volkswagen AG (VW) from all companies, the involved responsible persons are displayed.

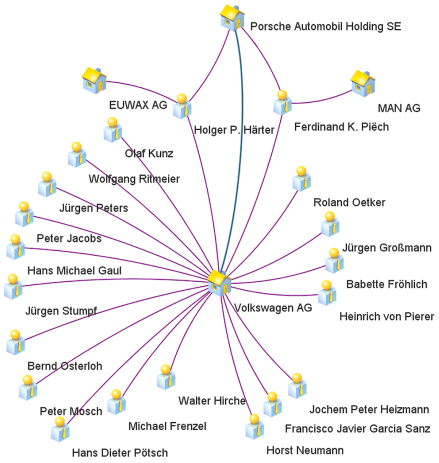

Fig. 5. Two directors of the board of VW are directors of Porsche, as well. The merger-relation holds between VW and Porsche (indicated by a thicker line between the companies).

is built containing entities ($v \in V$) and relations ($e \in E$) between entities. Its visualization is performed using the JUNG-Framework [14]. The human-computer interface allows users to select a company and move to the involved persons, from which the user may move to all the companies in which they play a role – thus browsing through the basic social network graph of economy. Figure 4 shows an example. Since the archives do not change their structure whenever the content changes, the database is easily updated.

3.2 Extracting the merger-relation from web documents

For monitoring the web of economy, the merger-relation is most interesting. To get a preselection of relevant documents the the web is crawled for information about the 30 DAX indexed German companies. Given a list of known company

names, the texts of the resulting websites are tagged in the IOB-scheme indicating "company"-entities. Only those sentences containing at least two company entities are selected for further processing. It is then the task of *relation extraction* to identify the true merger-relations between two companies. Of course, simple co-occurrence is not sufficient for this task. Note, that a sentence with three company names can include none, one, or two merger-relations. Hence, we applied our method described in Section 4. Details on the experiments are given in Section 5. Figure 5 shows an example of a found merger-relation.

4 Relation extraction with an enhanced composite kernel

We have implemented the state-of-the-art kernel method in Java, extending the kernel functions of SVM^{light} [8]. We also have developed an information extraction plug-in [9] for RapidMiner (formerly Yale) [12] including the composite kernel and all necessary preprocessing. When handing over the examples to the kernel functions, an example is split into the features for the linear kernel and into the tree for the tree kernel. When passing the tree to the kernel function, it may be pruned and enriched by new features.
We changed two aspects concerning the state-of-the-art composite kernels used for *relation extraction*:

- First, we widened the featureset used for the linear kernel.
- Second, we added semantic information to the tree kernel.

Features which contain words or word-parts of special positions in the sentence related to the entity's position showed to be useful for named entity recognition. However, for *relation extraction*, the position is no longer decisive. The information about the relation is spread all over the sentences, shows up at very different places, and can, hence, not be generalized. The clue verb *fusionieren*, for instance, may occur at various positions of the word sequence (see Figure 1). Especially for distinguishing between positive and negative *relation candidates*, the contextual information is not restricted to the words between the entities or to some words in front or behind the entities, as assumed by the feature set in [23]). In order to capture the influence of words that can act as an indicator for a relation we extract the word vector (containing just the word stems) of the complete sentence and add it to the linear features. In a separate experiment setting we use the features presented by [23], for comparison.
The second of our enhancements concerns the tree kernel. Figure 1 shows a parse tree of our corpus containing two entities (underlined solid) and the merger-indicating verbs (underlined dashed). It is easy to see that well-known subtrees for better *relation extraction* like shortest path-enclosing trees (SPT) will not work well in this context. But using the whole and unaltered parse tree will not work as well. The reason is, if a sentence contains positive and negative *relation candidates* the identical parse tree would be used for both *relation candidate*-types.
Using the context-sensitive parse tree of [24] is promising. But this approach

needs well-trained parsers which are still not available in an appropriate version for the German language. We therefore generalized the parse tree by adding syntactical information directly into the tree. First of all we marked the entities of the current *relation candidate* in the corresponding parse tree. In addition, we added semantic information into the parse tree by introducing extra nodes containing the word stems of the sentence at different depths.

Figure 6 shows four different types of parse trees used in our experimental

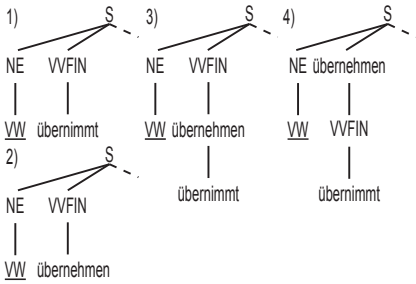

Fig. 6. Word stems at different depth-levels in the parse tree

settings. The first one (1)) is the original parse tree. In parse tree 2) we have replaced all terminals by their stems. Parse tree 3) is the tree after inserting the stem at depth 0. The depth is the depth of the stem in relation to the depth of the pre-terminal symbol. 4) is the tree after inserting the stem at depth 1. The word 'VW' has no stem, so nothing is inserted.

5 Experiments

Our companies corpus consists of 1698 sentences containing 3602 *relation candidates*. 2930 of these *relation candidates* are negative ones (being no merger-relation), and 672 *relation candidates* are true merger-relations. Only 98 of these 1689 sentences contain multiple *relation candidates* with different labels. Compared to other *relation extraction* datasets this distribution is very skewed and leads to the behavior described in the following Section. The ACE04 corpus for instance contains 2981 sentences out of which 1654 sentences contain at least a true relation and a negative candidate.

We produced several training sets with different attribute sets to compare our enhancements with the state-of-the-art composite kernel methods for *relation extraction*. All the training sets consist of all *relation candidates*, i.e., pairs of entities found in one sentence. Each example consists of a relation label, e.g. *merger* or *nomerger*, the syntactic tree of the sentence in which the arguments occur, and several features which are now described in detail.

The *baseline-featureset* contains the word-features proposed by [23]. These features are mainly based on words of the *relation candidate* entities or words

nearby in the sentence. For its use by the tree kernel, the feature set also contains the parse tree of the sentence the *relation candidate* is extracted from.

The *word-vector-featureset* contains just the word-vector of the sentence from which the *relation candidate* is extracted, and the parse tree.

The *big-word-vector-featureset* contains just the word-vector, the parse tree and the *baseline-featureset*.

The *stem-x-tree-featuresets* are equal to the *word-vector-featureset* but the parse tree contains the word stems inserted at depth-level x or as a replacement of the original terminal symbol.

The parse tree is given by running the Stanford parser [10] trained on the NEGRA corpus [16]. We applied 10-fold cross validation using the composite kernel with a parameter setting of $C = 2.4$, $m = 3$ and $\alpha = 0.6$ (see Section 2.2).

Performance Table 1 shows the performance of the state of the art method and the two versions of our new method. Table 2 shows the standard deviation of the performance measures in 10-fold cross validation.

Table 1. Performance of *relation extraction* on the companies corpus using 10-fold cross validation.

Featureset	Precision	Recall	F-meas.
baseline	33,47%	52,27%	38,64%
word-vector	36,41%	69,93%	45,45%
big-word-vector	36,83%	74,86%	**48,73%**
stem-replace-tree	31,46%	**76,03%**	44,08%
stem-0-tree	37,94%	47,90%	41,79%
stem-1-tree	**44,33%**	53,42%	47,51%
stem-2-tree	36,28%	62,91%	45,64%

Table 2. Standard deviation of the performance of *relation extraction*

Precision	Recall	F-meas.
3,88%	21,99%	8,88%
12,16%	11,64%	5,12%
5,15%	9,55%	**3,12%**
4,63%	**8,99%**	4,32%
6,29%	14,55%	9,07%
7,58%	9,89%	4.40%
3,95%	10,89%	4.62%

As can be seen, recall increases significantly using word vectors in the linear kernel and word stems in the tree kernel while at the same time the deviation decreases. Precision is best when semantic information in the tree is used at level 1. The best F-measure achieved by the big-word-vector is to be explained by the very few sentences containing a positive and a negative candidate of a relation. If sentences include either a positive or a negative example of a relation, the *relation extraction* is downgraded to sentence classification, where word vectors are a well suited representation. Hence, for *relation extraction*, the enhanced trees remain important.

6 Conclusions and Future Work

We proposed an economic network that is built up extracting semi-structured websites containing financial stock information.

The network – consisting of entities and relations between them – should be kept up to date automatically. Therefore we presented an enhancement to state-of-the-art *relation extraction* methods. Our enhancements take into account the problems German language faces in contrast to the well-examined English language.
To evaluate our method we extracted a German document corpus of the economic domain. We tagged all the firms in our corpus and extracted all possible *relation candidates*. We tested state-of-the-art *relation extraction* methods on our *relation extraction* corpus and compared the results with the results achieved by our enhancements.
Our enhanced composite kernel method achieves significantly better performance compared to the baseline. Although using just the linear kernel performs best, the usage of the composite kernel will be needed if the relations become more frequent and the number of relation-types becomes bigger.
Future work will implement better measures for the evaluation, so that sentence classification effects in *relation extraction* can properly be detected. Our approach should be evaluated on English benchmark datasets. Additionally our approach to add semantic information in the parse trees could be replaced by using dependency trees. But unfortunatelly the used library (stanford parser) just offers trained dependency parsers for English and Chinese. Using dependency trees therefore might be tested on English datasets.

References

1. R. C. Buncescu and R. J. Mooney. A shortest path dependency kernel for relation extraction. In *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 724–731, Vancouver, British Columbia, Canada, 2005. Association for Computational Linguistics.
2. R. C. Bunescu and R. J. Mooney. Subsequence kernels for relation extraction. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 171–178. MIT Press, 2006.
3. M. Collins and N. Duffy. Convolution kernels for natural language. In *Advances in Neural Information Processing Systems 14*, pages 625–632. MIT Press, 2001.
4. S. Debnath, N. Ganguly, and P. Mitra. Feature weighting in content based recommendation system using social network analysis. In *WWW 2008*. ACM Press, 2008.
5. P. Domingos and M. Richardson. Mining the network value of customers. In *Procs. KDD*, pages 57–66. ACM Press, 2001.
6. S. A. Golder, D. M. Wilkinson, and B. A. Huberman. Rhythms of social interaction: Messaging within a massive online network. In *Procs. 3rd Intl. Conf. on Communities and Technologies*, 2007.
7. D. Haussler. Convolution kernels on discrete structures. Technical report, University of California in Santa Cruz, Computer Science Dept., 1999.
8. T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Procs. Of European Conference on Machine Learning*, pages 137 – 142. Springer, 1989.

9. F. Jungermann. Information extraction with rapidminer. In W. Hoeppner, editor, *Proceedings of the GSCL Symposium 'Sprachtechnologie und eHumanities'*, pages 50–61. Universität Duisburg-Essen, Abteilung für Informatik und Angewandte Kognitionswissenschaft Fakultät für Ingenieurwissenschaften, 2009.
10. D. Klein and C. D. Manning. Fast extract inference with a factored model for natural language parsing. In *Proceedings of Advances in Neural Information Processing Systems*, 2002.
11. Linguistic Data Consortium. *The ACE 2004 Evaluation Plan*, 2004.
12. I. Mierswa, M. Wurst, R. Klinkenberg, M. Scholz, and T. Euler. Yale: Rapid prototyping for complex data mining tasks. In *Procs. 12th ACM SIGKDD Int. Conf. .Knowledge Discovery and Data Mining (KDD)*, 2006.
13. A. Moschitti. Efficient convolution kernels for dependency and constituent syntactic trees. In J. Fuernkranz, T. Scheffer, and M. Spiliopoulou, editors, *Procs. ECML*, pages 318 – 329. Springer, 2006.
14. J. O'Madadhain, D. Fisher, S. White, and Y.-B. Boey. The JUNG (java universal network/graph) framework. Technical Report Technical Report UCI-ICS 03-17, School of Information and Computer Science University of California, Irvine, CA 92697-3425, 2003.
15. J. Palau, M. Montaner, B. Lpez, and J. L. D. L. Rosa. Collaboration analysis in recommender systems using social networks. In *Procs. Cooperative Information Agents VIII*, pages 137–151. Springer, 2004.
16. W. Skut, B. Krenn, T. Brants, and H. Uszkoreit. An annotation scheme for free word order languages. In *Proceedings of the Fifth Conference on Applied Natural Language Processing (ANLP)-97*, Washington, DC, 1997.
17. S. V. N. Vishwanathan and A. J. Smola. Fast kernels for string and tree matching. In *NIPS*, pages 569–576, 2002.
18. D. Zelenko, C. Aone, and A. Richardella. Kernel methods for relation extraction. *Journal of Machine Learning Research*, 3:1083–1106, 2003.
19. M. Zhang, W. Che, A. T. Aw, C. L. Tan, G. Zhou, T. Liu, and S. Li. A grammar-driven convolution tree kernel for semantic role classification. In *Procs. 4th Annual Meeting of ACL*, pages 200 – 207, 2007.
20. M. Zhang, J. Zhang, J. Su, and G. Zhou. A composite kernel to extract relations between entities with both flat and structured features. In *Procs. 44th Annual Meeting of ACL*, pages 825–832, 2006.
21. S. Zhao and R. Grishman. Extracting relations with integrated information using kernel methods. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 419–426, Morristown, NJ, USA, 2005. Association for Computational Linguistics.
22. D. Zhou, S. A. Orshanskiy, H. Zha, and C. L. Giles. Co-ranking authors and documents in a heterogeneous network. In *7th IEEE ICDM*, 2007.
23. G. Zhou, J. Su, M. Zhang, and J. Zhang. Exploring various knowledge in relation extraction. In *ACL*, pages 427 – 434, 2005.
24. G. Zhou, M. Zhang, D. H. Ji, and Q. Zhu. Tree kernel-based relation extraction with context-sensitive structured parse tree information. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, 2007.