A Logic-Based Approach to Relation Extraction from Texts*

Tamás Horváth[1,2], Gerhard Paass[2], Frank Reichartz[2], and Stefan Wrobel[2,1]

[1] Dept. of Computer Science III, University of Bonn, Germany
[2] Fraunhofer IAIS, Schloss Birlinghoven, Sankt Augustin, Germany
{tamas.horvath,gerhard.paass,frank.reichartz,stefan.wrobel}@iais.fraunhofer.de

Abstract. In recent years, text mining has moved far beyond the classical problem of text classification with an increased interest in more sophisticated processing of large text corpora, such as, for example, evaluations of complex queries. This and several other tasks are based on the essential step of relation extraction. This problem becomes a typical application of learning logic programs by considering the dependency trees of sentences as relational structures and examples of the target relation as ground atoms of a target predicate. In this way, each example is represented by a definite first-order Horn-clause. We show that Plotkin's LGG operator can effectively be applied to such clauses and propose a simple and effective divide-and-conquer algorithm for listing a certain set of LGGs. We use these LGGs to generate binary features and compute the hypothesis by applying SVM to the feature vectors obtained. Empirical results on the ACE–2003 benchmark dataset indicate that the performance of our approach is comparable to state-of-the-art kernel methods.

1 Introduction

For a long time, text mining has mostly focused on document classification. In recent years, however, there has been an increased interest in more advanced processing of large text corpora. This problem is motivated by practical applications, e.g., in question answering, information retrieval, ontology learning, bioinformatics etc. In case of question answering, for instance, current search engines are not enough powerful for complex queries, such as, for example, *"find UN officials born in Africa"*. Obviously, the internal representation of texts in a search index as sequences of words is insufficient to recover semantics from unstructured text (e.g., the *"born in"* relation in the above example). *Relation extraction* is one of the essential steps towards more complex automatic text processing. It is concerned with the problem of detecting and classifying predefined semantic relations among m-tuples (typically between pairs) of entities in unstructured texts.

* This work was partially supported by the German Federal Ministry of Economy and Technology under the Theseus Project.

Presented at the 19th International Conference on Inductive Logic Programming (ILP'09).

Early approaches to relation extraction were based on patterns expressed usually as regular expressions for words with wildcards (see, e.g., [6]). The underlying hypothesis of this direction assumes that terms sharing similar linguistic contexts are connected by similar semantic relations. Various methods have been developed for this approach using noun phrase annotations as well as WordNet hypernyms (see, e.g., [3]), applying frequent itemset mining to extract word patterns for relation extraction (see, e.g., [1]), or defining various measures for pattern reliability and filtering incorrect instances using the web (see, e.g., [10]).

Another promising direction is based on the utilization of low-level syntactic parse trees, in particular, *dependency trees*, enhancing kernel methods with additional information on the syntactic structure of sentences. While feature-based methods are restricted to a limited number of (structural) features to be used, kernel-based methods offer efficient solutions allowing the exploration of much larger (often exponential, or in some cases, infinite) set of characteristics of trees in polynomial time, without the need of explicit computation of the features. Examples of this approach include, e.g., the dependency tree kernels [4, 12] inspired by string kernels. In [2] another kernel for dependency trees is proposed for binary target relations which is based on the similarity between nodes on the shortest path of the dependency tree that connect the corresponding entities. All these kernels are used as input for a kernel classifier.

Similarly to [4], in this work we consider relation extraction as a supervised learning problem and transform the unstructured text to *dependency trees*. Regarding dependency trees as relational structures and examples of a particular m-ary target relation to be learnt as ground atoms of an m-ary target predicate with constants representing certain distinguished nodes of dependency trees, we can consider relation extraction as a typical application of learning logic programs. Applying Plotkin's least general generalization (LGG) operator [11], we generate a set of first-order definite non-recursive Horn-clauses satisfying certain frequency and consistency criterion, i.e., all these rules must cover at least a certain number of positive examples while implying at most a certain number of negative examples. In the generation of LGGs, we exploit the specific structure of dependency trees allowing polynomial time rule evaluation defined by θ-subsumption and the fact that the LGG is a closure operator on the power set of the instance space over the target predicate. Using these rules, we generate a binary feature vector for each example and, applying support vector machines to these feature vectors, find a hypothesis separating the positive and negative examples.

Our first experiments on the ACE–2003 benchmark corpus [9] clearly indicate that the above approach compares well to state-of-the-art methods [2, 4]. Furthermore, in contrast to other approaches restricted to the special case of binary target relations [2], our approach is applicable to arbitrary arity, i.e., to unary target relations, as well as to arities greater than 2. Preliminary results with various strategies enhancing dependency trees and the generalization operator by semantic features (e.g., by topic model scores for term disambiguation)

also suggest that the predictive performance of our approach can further be improved.

2 The Method

In this section we briefly describe the main steps and the most important algorithmic features of the logic-based relation extraction method proposed in this work. Due to space limitations, we omit the discussion of several technical issues in this extended abstract.

Data Preprocessing Given a set of sentences annotated with respect to the target relation, in a preprocessing step we first compute a *dependency tree* for each sentence. A dependency tree is a labeled rooted directed tree representing the grammatical dependencies among the words of a sentence. Thus, it captures a low-level syntactic structure of sentences. For the generation of dependency trees from sentences, we employ the `Stanford Parser`[3], a state-of-the art natural language parser. Since our aim is to detect semantic relationships among entities in a sentence, we merge the nodes of the dependency tree into single artificial nodes that define the same entities. As an example, the two nodes corresponding to "Barack Obama" will be merged into a single node labeled by "Barack_Obama". Thus, in the remainder of this work we assume without loss of generality that all entities are represented by single nodes in the dependency trees.

First-Order Logic Representation An important feature of the dependency trees obtained in this way is that there is an injective mapping from the set of entities in a sentence to the set of nodes in the corresponding dependency tree. Detecting particular instances of a given m-ary target relation thus corresponds to the problem of classifying (ordered) m-tuples of entity nodes in dependency trees. Notice that, in contrast to other approaches restricted to the special case of binary target relations, i.e., $m = 2$ (see, e.g., [2]), our approach is applicable to arbitrary arity (also to unary target relation).

We make use of this feature of dependency trees and the fact that dependency trees can be considered as *relational structures* in the standard natural way; the edges of the trees can be represented by a single binary predicate, while the labels by unary predicates. Furthermore, the instances of the target relation to be learnt can be considered as the instances of an unknown target predicate. This representation allows us to represent the examples (i.e., +/- labeled instance) of the target relation by ground non-recursive definite Horn-clauses as follows: The head of the Horn-clause is formed by the example and the body is composed of the ground literals representing the dependency trees.

[3] nlp.stanford.edu/software/lex-parser.shtml

Bottom-Up Generalization Using this representation, we define the generalization of a set of positive examples by Plotkin's least general generalization operator on clauses [11]. In our case, this operator is equivalent to the tensor (or weak direct) product of the labeled trees [7] corresponding to the bodies of the clauses representing the examples. We recall that the tensor product of the directed graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ is a directed graph $G = (V, E)$ such that $V = V_1 \times V_2$ and $E = \{((u_1, u_2), (v_1, v_2)) \in V \times V : (u_1, v_1) \in E_1 \text{ and } (u_2, v_2) \in E_2\}$.

Applying Plotkin's ordinary LGG to labeled graphs G_1 and G_2, we get that a product vertex in G is labeled by λ if and only if its components in G_1 and G_2 are labeled by λ. Otherwise, the product vertex remains unlabeled. In contrast to this definition, if the labels of the components in a product vertex are different, we generalize them by their least common ancestor in the WordNet. We note that, due to semantic ambiguity, this least common ancestor is not always unique; in this case we apply disambiguation techniques. It holds that this generalization of labels (or equivalently, unary predicates) remains equivalent to the ordinary LGG operator by labeling the nodes in the dependency trees by all of their more general categories in the WordNet.

One can easily see that forests are closed under tensor product implying that the graphs in the bodies of the LGGs are also forests. Clearly, an LGG of a set of positive examples can be a good generalization only when the product vertices corresponding to the arguments in the head all belong to the same tree in the body's forest. If this is the case, we keep only the tree containing the distinguished product vertices; otherwise, the LGG is considered to be illegal. This reduction significantly decreases the size of the LGG.

Rule Evaluation Since the LGGs obtained are neither recursive nor self-resolvent, implication for this case becomes equivalent to θ-subsumption [5], which, in turn, is equivalent to *relational homomorphism* [8]. Although homomorphism between relational structures is computationally intractable, in our case it can be decided in polynomial time because homomorphism from labeled trees into graphs can be decided in polynomial time. Thus, we can decide in polynomial time whether or not an LGG implies (i.e., covers) an m-tuple of entities in a sentence. The above remarks also imply that redundant literals can efficiently be eliminated from the LGG, as this step is also based on deciding relational homomorphism. Though the size of the reduced LGG obtained can still exponentially grow with the number of examples, our experimental results clearly indicate that it remains small in practice.

Rule Enumeration and Application The LGG can also be considered as a function on the power set of the instance space corresponding to the target predicate because it assigns a set of m-tuples to a set of m-tuples. One can easily check that this function is extensive, monotone, and idempotent. Thus, the LGG is a closure operator on the power set of the instance space. We exploit this fact and, using a simple divide-and-conquer algorithm, generate a certain subset of the closed sets with respect to the LGG. More precisely, we generate only such

Method	Precision	Recall	F_{micro}	F_{macro}
logic-based	68.2%	42.3%	52.2%	45.3%
shortest path kernel [2]	65.5%	53.8%	52.5%	–
subtree kernel [4]	67.1%	35.0%	45.8%	–

Table 1. Table of results on relations between name mentions.

LGGs that are *frequent* with respect to the positive examples and *infrequent* with respect to the negative examples.

For each closed set we take its reduced LGG representation as described above. This set of LGGs is then used as Boolean features for the examples of the target relation. In this way, we obtain a Boolean feature vector for every training examples and learn a separating hyperplane by using SVM.

3 Empirical Results

In this section we compare our method with the approaches proposed in [2, 4] on the publicly available benchmark dataset ACE–2003[4] [9]. The ACE–2003 corpus was created during the Automatic Content Extraction conference series to evaluate and compare different approaches to information extraction from natural language texts. This corpus consists of 519 natural language text documents from different sources. The documents are all news related and consist of newspaper articles, newswire texts, and transcripts of broadcast news. There are 9256 sentences in the corpus with an average of 18 words per sentence. The entities and relations among them were annotated by experts. The texts are annotated according to the following 5 top level target relations (we also give the number of their occurrences): *role* (732), *part* (265), *near* (44), *social* (55), and *at* (481). These relations are then further refined into 24 subrelations. The relation $role(e_1, e_2)$ for example constitutes that entity e_1 has some "role" at entity e_2 (e.g., e_1 ="Barack_Obama" and e_2 ="USA").

We have performed a 5-fold cross-validation and calculated both the averaged macro (per class) and micro (per instance) evaluation measures, as we have a multi-class learning problem. The results of our logic-based method are depicted in Table 1 together with those reported for the shortest path kernel [2] and the subtree kernel [4]. While our method outperforms the subtree kernels both in precision and recall, the shortest path kernel only in precision. Though the shortest path kernel has better recall, we note that our method is more general than the shortest path kernel, as it is not restricted to binary target relations. Since no macro evaluation measures are reported in [2, 4], we are working on the reimplementation of these methods.

[4] Available from LDC (www.ldc.upenn.edu) as corpus LDC2003T11

4 Summary

We have proposed a logic-based method for relation extraction from natural language texts. Our method is based on transforming examples into definite Horn-clauses by considering dependency trees as relational structures. The special structure of the clauses obtained enable an effective applications of Plotkin's least general generalization operator. Since this operator is a closure operator on the power set of the instance space, we generate them by using a simple divide-and-conquer algorithm. The set of rules enumerated are then used to calculate binary feature vectors for the examples from which the final hypothesis is obtained by applying support vector machines. Empirical results on a popular benchmark dataset indicate that the performance of our method is comparable with state-of-the-art methods. It is important to emphasize that, in contrast to several other approaches, our method is not restricted to binary target relations.

References

1. S. Blohm and P. Cimiano. Scaling up pattern induction for web relation extraction through frequent itemset mining. In *Proc. of the KI 2008 Workshop on Ontology-Based Information Extraction Systems*, 2008.
2. R. C. Bunescu and R. J. Mooney. A shortest path dependency kernel for relation extraction. In *Proc. of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pp. 724–731. Association for Computational Linguistics, Morristown, NJ, USA, 2005.
3. P. Cimiano, A. Pivk, L. Schmidt-Thieme, and S. Staab. Learning taxonomic relations from heterogeneous evidence. In *Ontology Learning from Text: Methods, evaluation and applications*, IOS Press, 2005.
4. A. Culotta and J. Sorensen. Dependency tree kernels for relation extraction. In *Proc. of the 42nd Annual Meeting on Association for Computational Linguistics*, page 423. Association for Computational Linguistics, Morristown, NJ, USA, 2004.
5. G. Gottlob. Subsumption and implication. *Information Processing Letters*, 24(2):109–111, 1987.
6. M. Hearst. Automatic acquisition of hyponyms from large text corpora.. In *Proc. of the 15th Int. Conf. on Computational Linguistics*, 1992.
7. T. Horváth and G. Turán. Learning logic programs with structured background knowledge. *Artificial Intelligence*, 128(1–2):31–97, 2000.
8. P. Kolaitis and M. Y. Vardi. Conjunctive-query containment and constraint satisfaction. *Journal of Computer and System Sciences*, 61:302–332, 2000.
9. A. Mitchell, S. Strassel, M. Przybocki, J. Davis, G. Doddington, R. Grishman, A. Meyers, A. Brunstein, L. Ferro, and B. Sundheim. Ace-2 version 1.0. Linguistic Data Consortium, Philadelphia, 2003.
10. P. Pantel and M. Pennacchiotti. Espresso: leveraging generic patterns for automatically harvesting semantic relations. In *Proc. of the 21st Int. Conf. on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics Year of Publication: 2006*, pages 113 – 120, 2006.
11. G. Plotkin. A note on inductive generalisation. In *Machine Intelligence 5*, pages 153–163. Elsevier North Holland, New York, 1970.
12. D. Zelenko, C. Aone, and A. Richardella. Kernel methods for relation extraction. *JMLR*, 3:10831106., 2003.