Relation Extraction Using Support Vector Machine

Gumwon Hong

University of Michigan,
Ann Arbor, MI 48109
gwhong@umich.edu

Abstract. This paper presents a supervised approach for relation extraction. We apply Support Vector Machines to detect and classify the relations in Automatic Content Extraction (ACE) corpus. We use a set of features including lexical tokens, syntactic structures, and semantic entity types for relation detection and classification problem. Besides these linguistic features, we successfully utilize the distance between two entities to improve the performance. In relation detection, we filter out the negative relation candidates using entity distance threshold. In relation classification, we use the entity distance as a feature for Support Vector Classifier. The system is evaluated in terms of recall, precision, and F-measure, and errors of the system are analyzed with proposed solution.

1 Introduction

The goal of Information Extraction (IE) is to pull out pieces of information that are salient to the user's need from large volume of text. With the dramatic increase of World Wide Web, there has been growing interest in extracting relevant information from large textual documents. The IE tasks may vary in detail and reliability, but two subtasks are very common and closely related: named entity recognition and relation extraction. Named entity recognition identifies named objects of interest such as person, organizations or locations. Relation extraction involves the identification of appropriate relations among these entities. Examples of the specific relations are *employee-of* and *parent-of*. *Employee-of* relation holds between a particular person and a certain organization and *parent-of* holds between a father and his child. According to Message Understanding Conferences (MUC), while named entities can be extracted with 90% or more in F measure with a state of the art system, relation extraction was not so successful [8].

In this paper we propose a supervised machine learning approach for relation extraction. The goal of this paper is to investigate an empirical method to find out useful features and experimental procedure to increase the performance of relation extraction. We divide the extraction task into two individual subtasks: relation detection and relation classification. Relation detection is involved in identifying from every pair of entities positive examples of relations which can fall into one of many relation categories. In relation classification, we assign a specific class to each detected relation. To each task, we apply distinct linguistic features ranging from lexical tokens to syntactic structures as well as the semantic type information of the entities. We also applied the distance between two entities to make the detection problem easier and to increase the performance of both the relation detection and classification. We apply Support Vector Machines (SVMs) partly because it represents the state of the art performance for many classification tasks [3]. As we will discuss in section 3 about features, we are working on very high dimensional feature space, and this often leads to overfitting. Thus, the main reason to use SVMs is that this learning algorithm has a robust rationale for avoiding overfitting [11].

This paper is structured in the following manner. In section 2, we survey the previous work on relation extraction with emphasis on supervised machine learning approaches. In section 3, the problem formalization, the description of dataset and the feature extraction methods will be discussed. In section 4, we conduct a performance evaluation of the proposed approach and error analysis on ACE dataset. Finally, in section 5, a conclusion and discussion of future work will be followed.

2 Related Work

Relation extraction was formulated as part of Message Understanding Conferences (MUC) [8], which has focused on a series of information extraction tasks, including analyzing free text, recognizing named entities, and identifying relations of a specified type. Work on relation extraction over the last two decades has progressed from linguistically unsophisticated models to the adaptation of NLP techniques that use shallow parsers or full parsers and complicated machine learning methods.

[7] addressed the relation extraction problem by extending the statistical parsing model which simultaneously recovers syntactic structures, named entities, and relations. They first annotated sentences for entities, descriptors, coreference links and relation links, and trained the sentences from Penn Treebank. Then they applied to new training sentences and enhanced the parse trees to include the IE information. Finally they re-retrained the parser on newly enriched training data.

More recently, [12] introduced the kernel methods to extract relations from 200 news articles from different publications. The kernels are defined over shallow parse representations of text and computed through a dynamic programming fashion. They generated relation examples from shallow parses for two relations: person-affiliation and organization-location. Performance of the kernel methods was compared with feature-based methods and it showed that kernels have promising performance.

[4] extended work in [12] to estimate kernel functions between augmented dependency trees. Their experiment has two steps. Relations were first detected and then classified in cascading manner. However, the recall was relatively low because many positive examples of relations were dropped during detection phase. Detecting relations is hard job for kernel methods because, in detection, all negative examples are heterogeneous and it is difficult to use similarity function.

[13] proposed weakly supervised learning algorithm for classifying relations in ACE dataset. He introduced feature extraction methods for his supervised approach as well as a bootstrapping algorithm using random feature projection (called BootProject) on top of SVMs. He compared BootProject with supervised approach, and showed that BootProject algorithm reduced much work needed for labeling training data without decreasing the performance.

Our approach to relation extraction adapts some feature extraction methods from [13] but differs from [13] in two important aspects. First, instead of only classifying relations by assuming all candidate relations are detected, we perform relation detection before classifying relations by regarding every pair of entities in a sentence as a target of classification. Second, we used total 7652 (6140 for training and 1512 for development testing) relation examples in ACE dataset, but in [13] only 4238 relations were used for training and 1022 for testing. We increase the performance by using more training data and reducing errors in data processing stage.

3 Data Processing

3.1 Problem Definition

Our task is to extract relations from unstructured natural language text. Because we determine the relationship for every pair of entities in a sentence, our task is formalized with a standard classification problem as follows:

$$(e1\ ,e2,\ s) \rightarrow r$$

where *e1* and *e2* are two entities existing in sentence *s*, and *r* is a label of the relation. Every pair of entities in a sentence can be a relation or not, so we call the triple *(e1, e2, s)* a relation candidate. With the possible set of values of *r*, there can be at least three tasks.

1. **Detection only:** For each relation candidate, we predict whether it constitutes an actual relation or not. In this task, a label *r* can be either +1 (two entities are related) or -1 (not related).
2. **Combined detection and classification:** For each relation candidate, we perform N+1 way classification, where N is the number of relation types. Five relation types are specified in next subsection. The additional class is -1 (two entities are not related).
3. **Classification only:** For each relation, we perform N way classification. In this task, we assume that all relation candidates in test data are manually labeled with N+1 way classification. We only consider the classifier's performance on the N relation types.

We use a supervised machine learning technique, specifically Support Vector Machine classifiers, and we empirically evaluate the performance in terms of recall, precision and F measure. To make the problem more precisely, we constrain the task of relation extraction with following assumptions:

1. Entities should be tagged beforehand so that all information regarding entities is available when relations are extracted.
2. Relations are binary, i.e., every relation takes exactly two primary arguments.
3. The two arguments of a relation, i.e., an entity pair, should explicitly occur within a sentence.
4. Evaluation is performed over five limited types of relations as in Table 1.

A relation is basically an ordered pair, thus "**Sam** was flown to **Canada**" contains the relation AT(Sam, Canada) but not AT(Canada, Sam). However, 7 relations in NEAR (relative location) and SOCIAL (associate, other-personal, other-professional, other-relative, sibling, and spouse) types are symmetric. For example, the sentences such as "**Bill** is the neighbor of **Sarah**." and "**The park** is two blocks from **Walnut Street**" do not distinguish the order of entities.

Table 1. Relation types and their subtypes in ACE 2 corpus

AT	BASED-IN, LOCATED, RESIDENCE
NEAR	RELATIVE-LOCATION
PART	OTHER, PART-OF, SUBSDIARY
ROLE	AFILIATE-PARTNER, CITIZEN-OF, CLIENT, FOUNDER, GENERAL-STAFF, MANAGEMENT, MEMBER, OTHER, OWNER
SOCIAL	ASSOCIATE, GRANDPARENT, OTHER-PERSONAL, OTHER-PROFESSIONAL, OTHER-RELATIVE, PARENT, SIBLING, SPOUSE

3.2 Data Set

We extract relations from the Automatic Content Extraction (ACE) corpus, specifically version 1.0 of the ACE 2 corpus, provided by the National Institute for Standards and Technology (NIST). ACE corpora consist of 519 annotated text documents assembled from a variety of sources selected from broadcast news programs, newspapers, and newswire reports [1].

According to the scope of the LDC ACE program 1, current research in information extraction has three main objectives: Entity Detection and Tracking (EDT), Relation Detection and Characterization (RDC), and Event Detection and Characterization (EDC). This paper focuses on the second sub-problem, RDC. For example, we want to determine whether a particular person is at certain location, based on the contextual evidence. According to the RDC guideline version 3.6, Entities are limited to 5 types (PERSON, ORGANIZATION, GEO-POLITICAL ENTITY (GPE), LOCATION, and FACILITY), and relations are also classified into 5 types:

Role: Role relations represent an affiliation between a Person entity and an Organization, Facility, or GPE entity.
Part: Part relations represent part-whole relationships between Organization, Facility and GPE entities.
At: At relations represent that a Person, Organization, GPE, or Facility entity is located at a Location entity.
Near: Near relations represent the fact that a Person, Organization, GPE or Facility entity is near (but not necessarily "at") a Location or GPE entity.
Social: Social relations represent personal and professional affiliations between Person entities.

Table 1 lists the 5 major relation types and their subtypes. The numbers in the bottom row indicate the number of instances in training data and development testing data. We do not classify the 24 subtypes of relations due to the sparse instances of many subtypes.

Table 2. Breakdown of the ACE dataset (#f and #r means number of files and number of relations respectively)

Training data		Held-out data		Test data	
#f	# r	#f	# r	#f	# r
325	4628	97	1512	97	1512

Table 2 shows the breakdown of the ACE dataset for our task. Training data takes about 60% of all ACE corpora, and each held-out data and test data contains about 20% of the whole distribution in the number of files as well as in the number of relations.[1]

3.3 Data Processing

We start with entity marked data and the first step is to detect the sentence boundary using the sentence segmenter provided by DUC competition[2]. Performance of the sentence segmentater is crucial for relation detection because the number of entity pairs combinatorially increases as a sentence grows in size. The original DUC segmenter fails to detect sentence boundary if the last word of a sentence is a named entity and annotated with brackets, i.e., '<' and '>'. We modified the DUC segmenter by adding simple rules where we mark sentence boundary if every bracket is followed by '?', '!', or '.'. At this step, we drop the sentences if the sentence has less than two entities.

Sentence parsing is then performed using Charniak parser [6] on these target sentences in order to obtain the necessary linguistic information. To facilitate the feature extraction, parse trees are converted into chunklink format [2]. Chunklink format contains the same information as the original parse tree, but it provides for each word the chunk to which it belongs, its grammatical function, grammatical structure hierarchy, and trace information.

3.4 Features

We are ready to introduce our feature sets. The following features include lexical tokens, part of speech, syntactic features, semantic entity types, and the distance between two entities (we will say "entity distance" to refer to distance between two entities in the remaining part of this paper). These features are selectively used for three tasks defined in section 3.1.

1. Words. Lexical token of both entity mentions[3] *as well as all the words between them. If two or more words constitute an entity, each individual word in an entity is a separate feature.*

2. Part of speech. Part of speech corresponding to each word feature described above.

[1] Training data and held-out data together are called "train", and test data is called "devtest" in the ACE distribution.

[2] http://duc.nist.gov/past_duc/duc2003/software/

[3] Entities are objects and they have a group of mentions, and the mentions are textual references to the objects.

3. Entity type. Entities have one of five types (person, location, organization, geo-political entity, or facility).

4. Entity mention type. An entity mention can be named, nominal, or pronominal. Entity mention type is also called the level of a mention.

5. Chunk tag. A word is inside (I), outside (O) or in the beginning (B) of a chunk. A chunk here stands for a standard phrase. Thus, the word "George" at the beginning of NP chunk "George Bush" has B-NP tag. Similarly, the word "named" has I-VP tag in VP chunk "was named".

6. Grammatical function tag. If a word is the head word of a chunk, it has function of whole chunk. The other words in a chunk that are not the head have "NOFUNC" as their function. For example, the word "task" is head of NP chunk "the task" and it has grammatical function tag "NP", while "the" has "NOFUNC".

7. IOB chain. The syntactic categories of all the constituents on the path from the root node to the leaf node of a parse tree. For example, a word's IOB chain I-S/I-NP/B-PP means that it is inside a sentence, inside a NP chunk, and in the beginning of PP chunk.

8. Head word Path. The head word sequence in the path between two entity mentions in the parse tree. This feature is used after removing duplicate elements.

9. Distance. The number of words between two entity mentions.

10. Order. Relative position of two relation arguments.

Two entities and all words between them have separate features of chunk tags (5), grammatical function tags (6), and IOB chains (7), and these three features are all automatically computed by chunklink.pl. See [2] for further description of these three terminologies. We use the same method as in [13] to use the features described in 3, 5, 6, 7, and 10. The concept of the head word path is adapted from [10].

Let us take a sentence "**Officials** in **New York** took the task" for an example. Two entity mentions, officials and New York, constitute a relation AT(officials, New York). For the noun phrase "officials in New York", the corresponding parse tree is shown in Fig 2.

To compute the head word path, we need to know what a head word is. At the terminal nodes, the head word is simply the word from the sentence the node represents. At non-terminal nodes in the parse tree, "The head of a phrase is the element inside the phrase whose properties determine the distribution of that phrase, i.e. the environments in which it can occur." [9]

The head word path is computed in the following way. The head words of the two entity mentions are officials and York respectively. The path between these two head words is the sequence officials-NP-PP-NP-York which is in bold in Fig. 1. Then we change each non-terminal into its head word, and we get officials-officials-in-York-York. If we remove the duplicate elements, we get officials-in-York. Finally, we replace the two entity mentions with their entity mention types, resulting in PERSON-in-GPE.

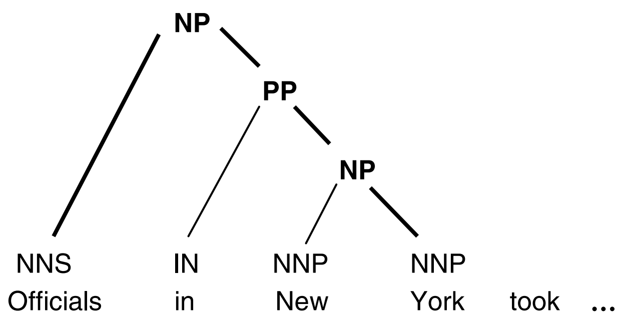

Fig. 1. Parse tree for the fragment "officials in New York". The path between two entity mentions is in bold.

Table 3 shows the features for the noun phrase "officials in New York". Note that all features are binary and they are assigned 1 or -1 depending on whether an example contains the feature or not. We could have *n*+1 possible distance features, where *n* is the entity distance threshold.

Table 3. Example of features for *Officials in New York.* e1 and e2 are two entities that constitute a relation AT(officials, New York).

Features	Example
Words	officials(e1), in, New(e2-1), York(e2-2)
Part-of-speech	NNS(e1), IN, NNP(e2-1), NNP(e2-2)
Entity type	officials: PERSON, New York: GPE
Entity mention type	officials: NOMINAL, New York: NAME
Chunk tag	B-NP, B-PP,B-NP, I-NP
Grammatical function tag	NP, PP, NOFUNC, NP
IOB chain	officials: B-S/B-NP/B-NP, in: I-S/I-NP/B-PP, New: I-S/I-NP/I-PP/B-NP York: I-S/I-NP/I-PP/I-NP
Head word path	PERSON-in-GPE
Distance	Distance=1
Order	e1 before e2

Besides the linguistic features, we apply entity distance to each task. When detecting actual relations, we use the entity distance to filter the unnecessary examples. To achieve the optimal performance, we set a threshold to keep the balance between recall and precision. We choose the threshold of entity distance based on the detection performance on held-out data. When classifying the individual relation types, the entity distance is also successfully used as a feature. We will discuss how the entity distance is utilized as a feature in the next section.

When performing combined detection and classification, we use two-staged extraction: we first detect relations in the same way as detection only task, and then perform classification on detected relations to predict the specific relation types. There are two important reasons that we choose this two-staged extraction.

First, we detect relations to increase the recall for whole extraction task. Because every combination of two entities in a sentence can construct a relation, we have far more negative examples of relation[4] in our training data. If we perform single-staged N+1 classification, the classifier is more likely to identify a testing instance as a non-relation because of the disproportionate size in samples which often decreases the performance. Thus in detection stage (or detection only task), we perform binary classification by enriching our training samples by assuming 5 types of relations to be one positive class.

Second, before performing relation classification, we can reduce a significant number of negative examples with only limited sacrifice of the detection performance by using a filtering technique with entity distance threshold. Fig. 2 shows the positive and negative examples distributed over the entity distances in training data. As we can see from the positive example curve (lower curve in the graph), the number of positive examples decreases as the entity distance exceeds 1. Positive examples do not exist where the entity distance is more than 32, while negative examples can have entity distances more than 60.

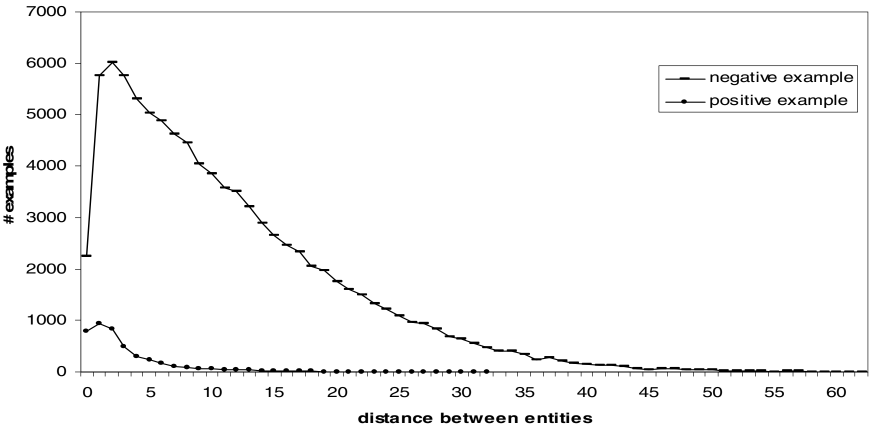

Fig. 2. Distribution of samples over entity distances in training data

We can also discover that almost 99% of the positive examples are distributed below the entity distance 20. If we restrict an entity distance threshold to 5, that is if we remove samples whose entity distances are more than 5, then whereas 71% of negative examples are dropped out, we can still maintain 80% of the positive examples. As can be seen in the graph, negative relations take more than 95% of whole entity pairs. This class imbalance makes the classifier less likely to identify a testing instance as a relation. The threshold makes relation detection easier because we can remove the large portion of negative examples, and we can get the result in reasonable time.

[4] We have total 6551 positive relation examples out of 131510 entity pairs in training data.

In our experiment we tested the following 3 sets of features:

F1: words + entity type + entity mention type + order
F2: F1 + all syntactic features (2, 5, 6, 7 and 8)
F3: F2 + distance

F1 is the set of simple features that can be obtained directly from the ACE corpus without applying any NLP techniques. Features in F2 require sentence boundary detection and parsing to acquire the syntactic structure of a sentence. F3 is used to determine whether an entity distance can be successfully used for our task.

4 Experiments and Error Analysis

We applied Support Vector Machines (SVMs) classifier to the task. More specifically, we used LIBSVM with RBF (Radial Basis Function) kernel [3]. SVMs attempt to find a hyperplane that split the data points with maximum margin, i.e., the greatest distance between opposing classes [11].

The system performance is usually reflected using the performance measures of information retrieval: precision, recall, and F-measure. Precision is the ratio of the number of correctly predicted positive examples to the number predicted positive examples. Recall is the ratio of the number of correctly predicted positive examples to the number of true positive examples. F-measure combines precision and recall as follows:

$$\text{F-measure} = 2 * \text{precision} * \text{recall} / (\text{precision} + \text{recall})$$

We report precision, recall, and F-measure for each experiment.

We use different features for relation detection and relation classification. In relation detection, which is either in detection only task or in the first stage of combined detection and classification, we do not use entity distance as a feature. Instead, we get rid of the negative examples of relations by setting a threshold value of the entity distances. The threshold is chosen in a way that we could achieve the best F-measure score of relation detection performance on held-out data with a classifier trained on training data. We have fixed the entity distance threshold to be 7 by this experiment, and this threshold enables us to remove 67% of the negative examples and keep 93% of the positive example. We will use this value in the remaining experiments.

In relation classification, which is either in the second stage of combined extraction or in classification only task, we use all the features described in Table 3. Entity distance is successfully used as a feature to classify the individual relation types. Let us take a relation type NEAR for an example to see why entity distance is useful. As shown in table 4, the average entity distance of NEAR relations is relatively greater than the average distances of any other types of relations.

Table 4. Average entity distance for 5 relation types in training data. The first row indicates the average entity distances over the relations whose entity distances are less than or equal to 7.

Relation	ROLE	PART	AT	NEAR	SOCIAL	ALL
Avg Entity distance (<= 7)	1.95	1.63	2.79	3.23	1.55	2.13
Avg Entity distance (all)	2.89	2.2	4.38	5.55	2.42	3.26

Table 5. Frequency of misclassification of relations (R: ROLE, A: AT, S: SOCIAL, N: NEAR, P: PART). Model was trained on training data with F2 features and tested on held-out data.

System	R	A	S	P	**P**	A	P	P	A	R	N	R	N
Gold	A	R	R	R	**N**	P	P	A	N	S	A	N	P
Frequency	46	34	29	23	**21**	19	12	11	7	6	1	1	1

This does not necessarily mean that we can choose NEAR when the entity distance in an example is close to 3.23. However, there is an interesting result about NEAR when we apply the classifier trained on training data to held-out data. Table 5 shows classification errors discovered after testing on held-out data. As shown in table 5, NEAR is misclassified as PART relatively often (almost 60% of the times). We also find by looking at samples in training data that only a small portion (about 5%) of the PART relations have entity distance more than 4, while more than 50% of NEAR relations have entity distance more than 4. This discovery suggests that we can prevent the misclassification by adding the distance features to original feature set to increase the performance.

Table 6. Detection only performance trained on training + held-out data and applied on test data (P: preprocessed with distance threshold = 7, N: no threshold is used)

Features	Precision	Recall	F-measure
F1 (N)	82.6	43.2	56.7
F2 (N)	69.4	62.1	65.5
F1 (P)	92.3	44.1	59.6
F2 (P)	77.8	69.3	73.3

Table 6 shows the performance of detection only task. The filtering process with distance threshold resulted in performance increase in all measurements.

As described in previous section, we divide the corpus into three different sets: training data, held-out data, and test data. We first build the classifier on training data, and applied the classifier to held-out data. In this experiment, we choose the threshold of entity distance until we get the optimal performance of relation detection on held-out data. We also count the misclassification of relations on held-out data. Next, we retrained the data on whole training set, i.e., training data plus held-out data, and then applied to the test data.

Table 7 shows the performance of combined detection and classification task on test data. We can see in table 7 that the system using all features achieved the highest performance. We can also find that the system using all syntactic features performs better in F-measure than the system without syntactic features. However, the features which do not require any language processing (F1) achieved relatively high precision compared to F2 and F3. When entities are very close, sometimes syntactic features would not be of much help. For example, relations such as "a **town** west of **Jerusalem**", "**park** outside **Paris**", "**his friend/wife/brother**" are highly dependent on the lexical feature, i.e., "west of", "outside", and "friend/wife/brother" are the most important clue to determine the class of the relations. Finally, as we previously discussed, F3 is always better, in all kinds of measurements, than F1 and F2. This proves that our system certainly benefits from the distance feature.

Table 7. Combined detection and classification performance. Model is trained on training data + held-out data and tested on test data (trained over 6140 relations and tested on 1512 relations)

Features	Precision	Recall	F-measure
F1	76.2	38.3	50.9
F2	65.2	49.7	56.4
F3	**68.8**	**51.4**	**58.8**

Table 8. Classification only performance. Model is trained on training data + held-out data and tested on test data (trained over 6140 relations and tested on 1512 relations).

Relation Type	Precision	Recall	F-measure
ROLE	85.7	84.3	85.0
PART	67.8	73.6	70.6
AT	81.5	82.9	82.2
NEAR	43.2	62.6	51.1
SOCIAL	74.9	88.2	81.0

We perform classification only task to compare the result with [13]. Table 8 shows the performance evaluation on 5 types of relation in test data. Direct comparison with [13] at each relation type is not a relevant process because the published result in [13] regarded a reduced set of relations (5,260 relations) as total relations tagged in the ACE data set. Nevertheless, our system performs better in F-measure in all relation types by capable of using much more relation examples and by utilizing entity distance feature.

5 Conclusion

We have presented a supervised approach for relation extraction where we apply Support Vector Machines to detect actual relations and to classify the specific types of those relations. We combine diverse lexical, syntactic and semantic features as well as entity distance.

Our system benefits from performing two-staged extraction as well as using the entity distance. In detection, we use binary classification to use more positive examples, and entity distance threshold also helps to remove large part of negative candidates for relations in reasonable time. Furthermore, entity distance is successfully used as a feature in classifying specific relations to increase the performance of the system.

The most immediate extension is to include semantic information such as semantic indexing and disambiguating the sense of entities. For example, as in table 5, SOCIAL relations are always confused with ROLE relations which often have similar lexical pattern. Moreover, NEAR and AT relations often have very similar syntactic structures. These examples do not benefit from lexical or syntactic features alone.

The next goal in this research will be to develop a classifier to apply other tasks of information extraction problem. We plan to integrate the entity recognition and relation extraction.

References

1. ACE. 2004. The NIST ACE evaluation website. http://www.nist.gov/speech/tests/ace/.
2. Buchholz, S.: The chunklink script (2000) http://ilk.uvt.nl/~sabine/chunklink/
3. Chang, C.-C. and Lin, C.-J.: LIBSVM: a library for support vector machines (2001) Software available at. http://www.csie.ntu.edu.tw/~cjlin/libsvm.
4. Culotta, A. and Sorensen, J.: Dependency tree kernels for relation extraction. Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics, Barcelona (2004)
5. Cortes, C. and Vapnik, V.: Supportvector networks. Machine Learning, (1995) 20(3):273–297
6. Charniak, E.: A maximum-entropy-inspired parser. Technical Report CS-99-12, Computer Scicence Department, Brown University (1999)
7. Miller, S., Crystal, M., Fox, H., Ramshaw, L., Schwartz, R., Stone, R., Weischedel, R., and The Annotation Group.: Algorithms that learn to extract information, bbn: Description of the sift system as used for muc-7. Technical report, BBN Technologies (2000)
8. National Institute of Standars and Technology. Proceedings of the 6th Message Undertanding Conference (MUC-7) (1998)
9. Sag, I., Wasow, T., and Bender, E.: Syntactic Theory: A Formal Introduction, volume 152 of CSLI Lecture Notes. CSLI Publications, Stanford, California, 2 edition (2003)
10. Singh, Natasha.: Syntactic features in relation extraction. MEng thesis. MIT. (2004)
11. Vapnik, V.: Statistical Learning Theory. John Wiley, (1998)
12. Zelenko, D., Aone, C., and Richardella, A.: Kernel methods for relation extraction. Journal of Machine Learning Research (2003) 1083–1106
13. Zhang, Z.: Weakly-supervised relation classification for information extraction. Proceedings of the Thirteenth ACM conference on Information and knowledge management. Washington D.C. (2004)