Using Corpus Statistics on Entities to Improve Semi-supervised Relation Extraction from the Web

Benjamin Rosenfeld
Information Systems
HU School of Business,
Hebrew University, Jerusalem, Israel
grurgrur@gmail.com

Ronen Feldman
Information Systems
HU School of Business,
Hebrew University, Jerusalem, Israel
ronen.feldman@huji.ac.il

Abstract

Many errors produced by unsupervised and semi-supervised relation extraction (RE) systems occur because of wrong recognition of entities that participate in the relations. This is especially true for systems that do not use separate named-entity recognition components, instead relying on general-purpose shallow parsing. Such systems have greater applicability, because they are able to extract relations that contain attributes of unknown types. However, this generality comes with the cost in accuracy. In this paper we show how to use corpus statistics to validate and correct the arguments of extracted relation instances, improving the overall RE performance. We test the methods on SRES – a self-supervised Web relation extraction system. We also compare the performance of corpus-based methods to the performance of validation and correction methods based on supervised NER components.

1 Introduction

Information Extraction (IE) is the task of extracting factual assertions from text. Most IE systems rely on knowledge engineering or on machine learning to generate the "task model" that is subsequently used for extracting instances of entities and relations from new text. In the knowledge engineering approach the model (usually in the form of extraction rules) is created manually, and in the machine learning approach the model is learned automatically from a manually labeled training set of documents. Both approaches require substantial human effort, particularly when applied to the broad range of documents, entities, and relations on the Web. In order to minimize the manual effort necessary to build Web IE systems, semi-supervised and completely unsupervised systems are being developed by many researchers.

The task of extracting facts from the Web has significantly different aims than the regular information extraction. The goal of regular IE is to identify and label all mentions of all instances of the given relation type inside a document or inside a collection of documents. Whereas, in the Web Extraction (WE) tasks we are only interested in extracting relation instances and not interested in particular mentions.

This difference in goals leads to a difference in the methods of performance evaluation. The usual measures of performance of regular IE systems are precision, recall, and their combinations – the breakeven point and F-measure. Unfortunately, the true recall usually cannot be known for WE tasks. Consequently, for evaluating the performance of WE systems, the recall is substituted by the number of extracted instances.

WE systems usually order the extracted instances by the system's confidence in their correctness. The precision of top-confidence extractions is usually very high, but it gets progressively lower when lower-confidence candidates are considered. The curve that plots the number of extractions against precision level is the best indicator of system's quality. Naturally, for a comparision be-

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 600–607,
Prague, Czech Republic, June 2007.

tween different systems to be meaningful, the evaluations must be performed on the same corpus.

In this paper we are concerned with Web RE systems that extract binary relations between named entities. Most of such systems utilize separate named entity recognition (NER) components, which are usualy trained in a supervised way on a separate set of manually labeled documents. The NER components recognize and extract the values of relation attributes (also called *arguments*, or *slots*), while the RE systems are concerned with patterns of contexts in which the slots appear. However, good NER components only exist for common and very general entity types, such as *Person*, *Organization*, and *Location*. For some relations, the types of attributes are less common, and no ready NER components (or ready labeled training sets) exist for them. Also, some Web RE systems (e.g., KnowItAll (Etzioni, Cafarella et al. 2005)) do not use separate NER components even for known entity types, because such components are usually domain-specific and may perform poorly on cross-domain text collections extracted from the Web.

In such cases, the values for relation attributes must be extracted by generic methods – shallow parsing (extracting noun phrases), or even simple substring extraction. Such methods are naturally much less precise and produce many entity-recognition errors (Feldman and Rosenfeld 2006).

In this paper we propose several methods of using corpus statistics to improve Web RE precision by validating and correcting the entities extracted by generic methods. The task of Web Extraction is particularly suited for the corpus statistics-based methods because of very large size of the corpora involved, and because the system is not required to identify individual mentions of the relations.

Our methods of entity validation and correction are based on the following two observations:

First, the entities that appear in target relations will often also appear in many other contexts, some of which may strongly discriminate in favor of entities of specific type. For example, assume the system encounters a sentence "*Oracle bought PeopleSoft.*" If the system works without a NER component, it only knows that "*Oracle*" and "*PeopleSoft*" are proper noun phrases, and its confidence in correctness of a candidate relation instance *Acquisition*(*Oracle*, *PeopleSoft*) cannot be very high. However, both entities occur many times elsewhere in the corpus, sometimes in strongly discriminating contexts, such as "*Oracle is a company that…*" or "*PeopleSoft Inc.*" If the system somehow learned that such contexts indicate entities of the correct type for the *Acquisition* relation (i.e., companies), then the system would be able to boost its confidence in both entities ("*Oracle*" and "*PeopleSoft*") being of correct types and, consequently, in (*Oracle*, *PeopleSoft*) being a correct instance of the *Acquisition* relation.

Another observation that we can use is the fact that the entities, in which we are interested, usually have sufficient frequency in the corpus for statistical term extraction methods to perform reasonably well. These methods may often correct a wrongly placed entity boundary, which is a common mistake of general-purpose shallow parsers.

In this paper we show how to use these observations to supplement a Web RE system with an entity validation and correction component, which is able to significantly improve the system's accuracy. We evaluate the methods using SRES (Feldman and Rosenfeld 2006) – a Web RE system, designed to extend and improve KnowItAll (Etzioni, Cafarella et al. 2005). The contributions of this paper are as follows:

- We show how to automatically generate the validating patterns for the target relation arguments, and how to integrate the results produced by the validating patterns into the whole relation extraction system.
- We show how to use corpus statistics and term extraction methods to correct the boundaries of relation arguments.
- We experimentally compare the improvement produced by the corpus-based entity validation and correction methods with the improvements produced by two alternative validators – a CRF-based NER system trained on a separate labeled corpus, and a small manually-built rule-based NER component.

The rest of the paper is organized as follows: Section 2 describes previous work. Section 3 outlines the general design principles of SRES and briefly describes its components. Section 4 describes in detail the different entity validation and correction methods, and Section 5 presents their experimental evaluation. Section 6 contains conclusions and directions for future work.

2 Related Work

We are not aware of any work that deals specifically with validation and/or correction of entity recognition for the purposes of improving relation extraction accuracy. However, the background techniques of our methods are relatively simple and known. The validation is based on the same ideas that underlie semi-supervised entity extraction (Etzioni, Cafarella et al. 2005), and uses a simplified SRES code. The boundary correction process utilizes well-known term extraction methods, e.g., (Su, Wu et al. 1994).

We also recently became aware of the work by Downey, Broadhead and Etzioni (2007) that deals with locating entities of arbitrary types in large corpora using corpus statistics.

The IE systems most similar to SRES are based on bootstrap learning: Mutual Bootstrapping (Riloff and Jones 1999), the DIPRE system (Brin 1998), and the Snowball system (Agichtein and Gravano 2000). Ravichandran and Hovy (Ravichandran and Hovy 2002) also use bootstrapping, and learn simple surface patterns for extracting binary relations from the Web.

Unlike these systems, SRES surface patterns allow gaps that can be matched by any sequences of tokens. This makes SRES patterns more general, and allows to recognize instances in sentences inaccessible to the simple surface patterns of systems such as (Brin 1998; Riloff and Jones 1999; Ravichandran and Hovy 2002).

Another direction for unsupervised relation learning was taken in (Hasegawa, Sekine et al. 2004; Chen, Ji et al. 2005). These systems use a NER system to identify frequent pairs of entities and then cluster the pairs based on the types of the entities and the words appearing between the entities. The main benefit of this approach is that all relations between two entity types can be discovered simultaneously and there is no need for the user to supply the relations definitions.

3 Description of SRES

The goal of SRES is extracting instances of specified relations from the Web without human supervision. Accordingly, the supervised input to the system is limited to the specifications of the target relations. A specification for a given relation consists of the *relation schema* and a small set of *seeds* – known true instances of the relation. In the full-scale SRES, the seeds are also generated automatically, by using a set of generic patterns instantiated with the relation schema. However, the seed generation is not relevant to this paper.

A relation schema specifies the name of the relation, the names and types of its arguments, and the arguments ordering. For example, the schema of the *Acquisition* relation

Acquisition(Buyer=ProperNP,
Acquired=ProperNP) ordered

specifies that *Acquisition* has two slots, named *Buyer* and *Acquired*, which must be filled with entities of type *ProperNP*. The order of the slots is important (as signified by the word "*ordered*", and as opposed to relations like *Merger*, which are "*unordered*" or, in binary case, "*symmetric*").

The baseline SRES does not utilize a named entity recognizer, instead using a shallow parser for exracting the relation slots. Thus, the only allowed entity types are *ProperNP*, *CommonNP*, and *AnyNP*, which mean the heads of, respectively, proper, common, and arbitrary noun phrases. In the experimental section we compare the baseline SRES to its extensions containing additional NER components. When using those components we allow further subtypes of *ProperNP*, and the relation schema above becomes

... (Buyer=Company, Acquired=Company) ...

The main components of SRES are the Pattern Learner, the Instance Extractor, and the Classifier. The Pattern Learner uses the seeds to learn likely patterns of relation occurrences. Then, the Instance Extractor uses the patterns to extract the candidate instances from the sentences. Finally, the Classifier assigns the confidence score to each extraction. We shall now briefly describe these components.

3.1 Pattern Learner

The *Pattern Learner* receives a relation schema and a set of seeds. Then it finds the occurences of seeds inside a large (unlabeled) text corpus, analyzes their contexts, and extracts common patterns among these contexts. The details of the patterns language and the process of pattern learning are not significant for this paper, and are described fully in (Feldman and Rosenfeld 2006).

3.2 Instance Extractor

The Instance Extractor applies the patterns generated by the Pattern Learner to the text corpus. In order to be able to match the slots of the patterns, the Instance Extractor utilizes an external shallow parser from the OpenNLP package (http://opennlp.sourceforge.net/), which is able to find all proper and common noun phrases in a sentence. These phrases are matched to the slots of the patterns. In other respects, the pattern matching and extraction process is straightforward.

3.3 Classifier

The goal of the final classification stage is to filter the list of all extracted instances, keeping the correct extractions, and removing mistakes that would always occur regardless of the quality of the patterns. It is of course impossible to know which extractions are correct, but there exist properties of patterns and pattern matches that increase or decrease the confidence in the extractions that they produce.

These properties are turned into a set of binary features, which are processed by a linear feature-rich classifier. The classifier receives a feature vector for a candidate, and produces a confidence score between 0 and 1.

The set of features is small and is not specific to any particular relation. This allows to train a model using a small amount of labeled data for one relation, and then use the model for scoring the candidates of all other relations. Since the supervised training stage needs to be run only once, it is a part of the system development, and the complete system remains unsupervised, as demonstrated in (Feldman and Rosenfeld 2006).

4 Entity Validation and Correction

In this paper we describe three different methods of validation and correction of relation arguments in the extracted instances. Two of them are "classical" and are based, respectively, on the knowledge-engineering, and on the statistical supervised approaches to the named entity recognition problems. The third is our novel approach, based on redundancy and corpus statistics.

The methods are implemented as components for SRES, called Entity Validators, inserted between the Instance Extractor and the Classifier. The result of applying Entity Validator to a candidate instance is an (optionally) fixed instance, with validity values attached to all slots. There are three validity values: *valid*, *invalid*, and *uncertain*.

The Classifier uses the validity values by converting them into two additional binary features, which are then able to influence the confidence of extractions.

We shall now describe the three different validators in details.

4.1 Small Rule-based NER validator

This validator is a small Perl script that checks whether a character string conforms to a set of simple regular expression patterns, and whether it appears inside lists of known named entities. There are two sets of regular expression patterns – for *Person* and for *Company* entity types, and three large lists – for known personal names, known companies, and "other known named entities", currently including locations, universities, and government agencies.

The manually written regular expression represent simple regularities in the internal structure of the entity types. For example, the patterns for *Person* include:

```
Person = KnownFirstName  [Initial]  LastName
Person = Honorific [FirstName] [Initial] LastName
Honorific = ("Mr" | "Ms" | "Dr" |…) ["."]
Initial = CapitalLetter ["."]
KnownFirstName = member of
                          KnownPersonalNamesList
FirstName = CapitalizedWord
LastName = CapitalizedWord
LastName = CapitalizedWord ["–"CapitalizedWord]
LastName = ("o" | "de" | …) "`"CapitalizedWord
      …
```

while the patterns for *Company* include:

```
Company = KnownCompanyName
Company = CompanyName CompanyDesignator
Company = CompanyName FrequentCompanySfx
KnownCompanyName = member of
                              KnownCompaniesList
CompanyName = CapitalizedWord +
CompanyDesignator = "inc" | "corp" | "co" | …
FrequentCompanySfx = "systems" | "software" | …
      …
```

The validator works in the following way: it receives a sentence with a labeled candidate entity of a specified entity type (which can be either *Person* or *Company*). It then applies all of the regular expression patterns to the labeled text and to its enclosing context. It also checks for membership in the lists of known entities. If a boundary is incorrectly placed according to the patterns or to the lists, it is fixed. Then, the following result is returned:

Valid, if some pattern/list of the right entity type matched the candidate entity, while there were no matches for patterns/lists of other entity types.

Invalid, if no pattern/list of the right entity type matched the candidate entity, while there were matches for patterns/lists of other entity types.

Uncertain, otherwise, that is either if there were no matches at all, or if both correct and incorrect entity types matched.

The number of patterns is relatively small, and the whole component consists of about 300 lines in Perl and costs several person-days of knowledge engineering work. Despite its simplicity, we will show in the experimental section that it is quite effective, and even often outperforms the CRF-based NER component, described below.

4.2 CRF-based NER validator

This validator is built using a feature-rich CRF-based sequence classifier, trained upon an English dataset of the CoNLL 2003 shared task (Rosenfeld, Fresko et al. 2005). For the gazetteer lists it uses the same large lists as the rule-based component described above.

The validator receives a sentence with a labeled candidate entity of a specified entity type (which can be either *Person* or *Company*). It then sends the sentence to the CRF-based classifier, which labels all named entities it knows – *Dates*, *Times*, *Percents*, *Persons*, *Organizations*, and *Locations*. If the CRF classifier places the entity boundaries differently, they are fixed. Then, the following result is returned:

Valid, if CRF classification of the entity accords with the expected argument type.

Invalid, if CRF classification of the entity is different from the expected argument type.

Uncertain, otherwise, that is if the CRF classifier didn't recognize the entity at all.

4.3 Corpus-based NER validator

The goal of building the corpus-based NER validator is to provide the same level of performance as the supervised NER components, while requiring neither additional human supervision nor additional labeled corpora or other resources. There are several important facts that help achieve this goal.

First, the relation instances that are used as seeds for the pattern learning are known to contain correct instances of the right entity type. These instances can be used as seeds in their own right, for learning the patterns of occurrence of the corresponding entity types. Second, the entities in which we are interested usually appear in the corpus with a sufficient frequency. The validation is based on the first observation, while the boundary fixing on the second.

Corpus-based entity validation

There is a preparation stage, during which the information required for validation is extracted from the corpus. This information is the lists of all entities of every type that appears in the target relations. In order to extract these lists we use a simplified SRES. The entities are considered to be unary relations, and the seeds for them are taken from the slots of the target binary relations seeds. We don't use the Classifier on the extracted entity instances. Instead, for every extracted instance we record the number of different sentences the entity was extracted from.

During the validation process, the validator's task is to evaluate a given candidate entity instance. The validator compares the number of times the instance was extracted (during the preparation stage) by the patterns for the correct entity type, and by the patterns for all other entity types. The validator then returns

Valid, if the number of times the entity was extracted for the specified entity type is at least 5, and at least two times bigger than the number of times it was extracted for all other entity types.

Invalid, if the number of times the instance was extracted for the specified entity type is less than 5, and at least 2 times smaller than the number of times it was extracted for all other entity types.

Uncertain, otherwise, that is if it was never extracted at all, or extracted with similar frequency for both correct and wrong entity types.

Corpus-based correction of entity boundaries

Our entity boundaries correction mechanism is similar to the known statistical term extraction techniques (Su, Wu et al. 1994). It is based on the assumption that the component words of a term (an entity in our case) are more tightly bound to each other than to the context. In the statistical sense, this fact is expressed by a high mutual information between the adjacent words belonging to the same term.

There are two possible boundary fixes: removing words from the candidate entity, or adding words from the context to the entity. There is a significant practical difference between the two cases.

Assume that an entity boundary was placed too broadly, and included extra words. If this was a chance occurrence (and only such cases can be found by statistical methods), then the resulting sequence of tokens will be very infrequent, while its parts will have relatively high frequency. For example, consider a sequence "*Formerly Microsoft Corp.*", which is produced by mistakenly labeling "*Formerly*" as a proper noun by the PoS tagger. While it is easy to know from the frequencies that a boundary mistake was made, it is unclear (to the system) which part is the correct entity. But since the entity (one of the parts of the candidate) has a high frequency, there is a chance that the relation instance, in which the entity appears, will be repeated elsewhere in the corpus and will be extracted correctly there. Therefore, in such case, the simplest recourse is to simply label the entity as *Invalid*, and not to try fixing the boundaries.

On the other hand, if a word was missed from an entity (e.g., "*Beverly O*", instead of "*Beverly O ' Neill*"), the resulting sequence will be frequent. Moreover, it is quite probable that the same boundary mistake is made in many places, because the same sequence of tokens is being analyzed in all those places. Therefore, it makes sense to try to fix the bounary in this case, especially since it can be done simply and reliably: a word (or several words) is attached to the entity string if both their frequencies and their mutual information are above a threshold.

5 Experimental Evaluation

The experiments described in this paper aim to confirm the effectiveness of the proposed corpus-based relation argument validation and correction method, and to compare its performance with the classical knowledge-engineering-based and supervised-training-based methods. The experiments were performed with five relations:

Acquisition(BuyerCompany, AcquiredCompany),
Merger(Company1, Company2),
CEO_Of(Company, Person),
MayorOf(City, Person),
InventorOf(Person, Invention).

The data for the experiments were collected by the KnowItAll crawler. The data for the Acquisition and Merger consist of about 900,000 sentences for each of the two relations. The data for the bound relations consist of sentences, such that each contains one of a hundred values of the first (bound) attribute. Half of the hundred are frequent entities (>100,000 search engine hits), and another half are rare (<10,000 hits).

For evaluating the validators we randomly selected a set of 10000 sentences from the corpora for each of the relations, and manually evaluated the SRES results generated from these sentences. Four sets of results were evaluated: the baseline results produced without any NER validator, and three sets of results produced using three different NER validators. For the *InventorOf* relation, only the corpus-based validator results can be produced, since the other two NER components cannot be adapted to validate/correct entities of type *Invention*.

The results for the five relations are shown in the Figure 1. Several conclusions can be drawn from the graphs. First, all of the NER validators improve over the baseline SRES, sometimes as much as doubling the recall at the same level of precision. In most cases the three validators show roughly similar levels of performance. A notable difference is the *CEO_Of* relation, where the simple rule-based component performs much better than CRF, which performs yet better than the corpus-based component. The *CEO_Of* relation is tested as bound, which means that only the second relation argument, of type *Person*, is validated. The *Person* entities have much more rigid internal structure than the other entities – *Companies* and *Inventions*. Consequently, the best performing of

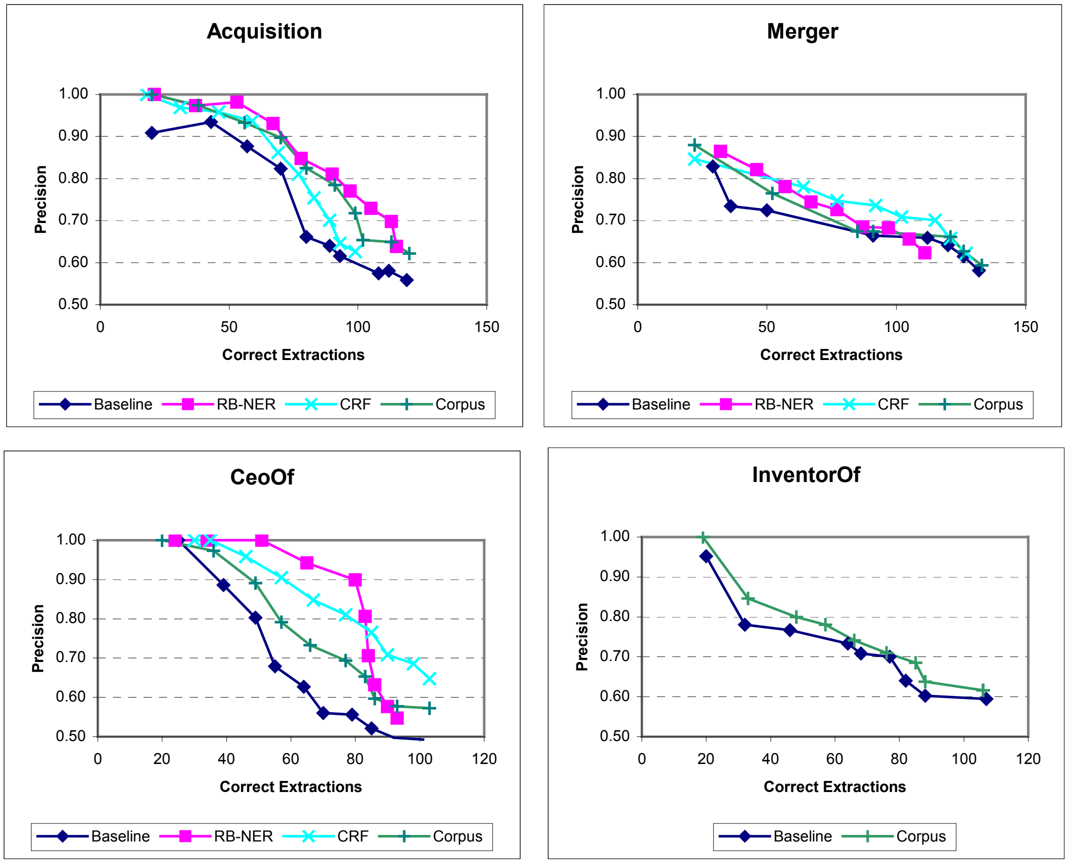

Figure 1. Comparison between Baseline-SRES and its extensions with three different NER validators: a simple Rule-Based one, a CRF-based statistical one, and a Corpus-based one.

the three validators is the rule-based, which directly tests this internal structure. The CRF-based validator is also able to take advantage of the structure, although in a weaker manner. The Corpus-based validator, however, works purely on the basis of context, entirely disregarding the internal structure of entities, and thus performs worst of all in this case. On the other hand, the Corpus-based validator is able to improve the results for the *Inventor* relation, which the other two validators are completely unable to do.

It is also of interest to compare the performance of CRF-based and the rule-based NER components in other cases. As can be seen, in most cases the rule-based component, despite its simplicity, outperforms the CRF-based one. The possible reason for this is that relation extraction setting is significantly different from the classical named entity recognition setting. A classical NER system is set to maximize the F_1 measure of all mentions of all entities in the corpus. A relation argument extractor, on the other hand, should maximize its performance on relation arguments, and apparently their statistical properties are often significantly different.

6 Conclusions

We have presented a novel method for validation and correction of relation arguments for the state-of-the-art unsupervised Web relation extraction system SRES. The method is based on corpus statistics and requires no human supervision and no additional corpus resources beyond the corpus that is used for relation extraction.

We showed experimentally the effectiveness of our method, which performed comparably to both simple rule-based NER and a statistical CRF-based NER in the task of validating *Companies*, and somewhat worse in the task of validating *Persons*, due to its complete disregard of internal structure of entities. The ways to learn and use this structure in an unsupervised way are left for future research.

Our method also successfully validated the *Invention* entities, which are inaccessible to the other methods due to the lack of training data.

In our experiments we made use of a unique feature of SRES system – a feature-rich classifier that assigns confidence score to the candidate instances, basing its decisions on various features of the patterns and of the contexts from which the candidates were extracted. This architecture allows easy integration of the entity validation components as additional feature generators. We believe, however, that our results have greater applicability, and that the corpus statistics-based components can be added to RE systems with other architectures as well.

References

Agichtein, E. and L. Gravano (2000). *Snowball: Extracting Relations from Large Plain-Text Collections*. Proceedings of the 5th ACM International Conference on Digital Libraries (DL).

Brin, S. (1998). *Extracting Patterns and Relations from the World Wide Web*. WebDB Workshop at 6th International Conference on Extending Database Technology, EDBT'98, Valencia, Spain.

Chen, J., D. Ji, C. L. Tan and Z. Niu (2005). *Unsupervised Feature Selection for Relation Extraction*. IJCNLP-05, Jeju Island, Korea.

Downey, D., M. Broadhead and O. Etzioni (2007). *Locating Complex Named Entities in Web Text*. IJCAI-07.

Etzioni, O., M. Cafarella, D. Downey, A. Popescu, T. Shaked, S. Soderland, D. Weld and A. Yates (2005). *Unsupervised named-entity extraction from the Web: An experimental study*. Artificial Intelligence 165(1): 91-134.

Feldman, R. and B. Rosenfeld (2006). *Boosting Unsupervised Relation Extraction by Using NER*. EMNLP-06, Sydney, Australia.

Feldman, R. and B. Rosenfeld (2006). *Self-Supervised Relation Extraction from the Web*. ISMIS-2006, Bari, Italy.

Hasegawa, T., S. Sekine and R. Grishman (2004). *Discovering Relations among Named Entities from Large Corpora*. ACL 2004.

Ravichandran, D. and E. Hovy (2002). *Learning Surface Text Patterns for a Question Answering System*. 40th ACL Conference.

Riloff, E. and R. Jones (1999). *Learning Dictionaries for Information Extraction by Multi-level Bootstrapping*. AAAI-99.

Rosenfeld, B., M. Fresko and R. Feldman (2005). *A Systematic Comparison of Feature-Rich Probabilistic Classifiers for NER Tasks*. PKDD.

Su, K.-Y., M.-W. Wu and J.-S. Chang (1994). A Corpus-based Approach to Automatic Compound Extraction. *Meeting of the Association for Computational Linguistics*: 242-247.