Learning Effective Surface Text Patterns
for Information Extraction

Gijs Geleijnse and Jan Korst
Philips Research Laboratories
Prof. Holstlaan 4, 5656 AA Eindhoven, The Netherlands
{gijs.geleijnse,jan.korst}@philips.com

Abstract

We present a novel method to identify effective surface text patterns using an internet search engine. Precision is only one of the criteria to identify the most effective patterns among the candidates found. Another aspect is frequency of occurrence. Also, a pattern has to relate diverse instances if it expresses a non-functional relation. The learned surface text patterns are applied in an ontology population algorithm, which not only learns new instances of classes but also new instance-pairs of relations. We present some first experiments with these methods.

1 Introduction

Ravichandran and Hovy (2002) present a method to automatically learn surface text patterns expressing relations between instances of classes using a search engine. Their method, based on a training set, identifies natural language surface text patterns that express some relation between two instances. For example, "was born in" proved to be a precise pattern expressing the relation between instances Mozart (of class 'person') and 1756 (of class 'year').

We address the issue of learning surface text patterns, since we observed two drawbacks of Ravichandran and Hovy's work with respect to the application of such patterns in a general information extraction setting.

The first drawback is that Ravichandran and Hovy focus on the use of such surface text patterns to answer so-called *factoid* questions (Voorhees, 2004). They use the assumption that each instance is related by $\mathcal{R}$ to exactly one other instance of some class. In a general information extraction setting, we cannot assume that all relations are functional.

The second drawback is that the criterion for selecting patterns, precision, is not the only issue for a pattern to be effective. We call a pattern effective, if it links many different instance-pairs in the excerpts found with a search engine.

We use an ontology to model the information domain we are interested in. Our goal is to populate an ontology with the information extracted. In an ontology, instances of one class can be related by some relation $\mathcal{R}$ to multiple instances of some other class. For example, we can identify the classes 'movie' and 'actor' and the 'acts in'-relation, which is a many-to-many relation. In general, multiple actors star in a single movie and a single actor stars in multiple movies.

In this paper we present a domain-independent method to learn effective surface text patterns representing relations. Since not all patterns found are highly usable, we formulate criteria to select the most effective ones. We show how such patterns can be used to populate an ontology.

The identification of effective patterns is important, since we want to perform as few queries to a search engine as possible to limit the use of its services.

This paper is organized as follows. After defining the problem (Section 2) and discussing related work (Section 3), we present an algorithm to learn effective surface text patterns in Section 4. We discuss the application of this method in an ontology population algorithm in Section 5. In Section 6, we present some of our early experiments. Sections 7 and 8 handle conclusions and future work.

2 Problem description

We consider two classes c_q and c_a and the corresponding non-empty sets of instances I_q and I_a. Elements in the sets I_q and I_a are instances of c_q and c_a respectively, and are known to us beforehand. However, the sets I do not have to be complete, i.e. not all possible instances of the corresponding class have to be in the set I.

Moreover, we consider some relation $\mathcal{R}$ between these classes and give a non-empty training set of instance-pairs $T_{\mathcal{R}} = \{(x, y) \mid x \in I_q \wedge y \in I_a\}$, which are instance-pairs that are known to be $\mathcal{R}$-related.

Problem: Given the classes c_q and c_a, the sets of instances I_q and I_a, a relation $\mathcal{R}$ and a set of $\mathcal{R}$-related instance-pairs $T_{\mathcal{R}}$, learn effective surface text patterns that express the relation $\mathcal{R}$.

Say, for example, we consider the classes 'author' and 'book title' and the relation 'has written'. We assume that we know some related instance-pairs , e.g. ('Leo Tolstoy', 'War and Peace') and ('Günter Grass', 'Die Blechtrommel'). We then want to find natural language phrases that relate authors to the titles of the books they wrote. Thus, if we query a pattern in combination with the name of an author (e.g. 'Umberto Eco wrote'), we want the search results of this query to contain the books by this author.

The population of an ontology can be seen as a generalization of a question-answering setting. Unlike question-answering, we are interested in finding all possible instance-pairs, not only the pairs with one fixed instance (e.g. all 'author'-'book' pairs instead of only the pairs containing a fixed author). Functional relations in an ontology correspond to factoid questions, e.g. the population of the classes 'person' and 'country' and the 'was born in'-relation. Non-functional relations can be used to identify answers to list questions, for example "name all books written by Louis-Ferdinand Céline" or "which countries border Germany?".

3 Related work

Brin identifies the use of patterns in the discovery of relations on the web (Brin, 1998). He describes a website-dependent approach to identify hypertext patterns that express some relation. For each web site, such patterns are learned and explored to identify instances that are similarly related. In (Agichtein and Gravano, 2000), such a system is combined with a named-entity recognizer.

In (Craven et al., 2000) an ontology is populated by crawling a website. Based on tagged web pages from other sites, rules are learned to extract information from the website.

Research on named-entity recognition was addressed in the nineties at the Message Understanding Conferences (Chinchor, 1998) and is continued for example in (Zhou and Su, 2002).

Automated part of speech tagging (Brill, 1992) is a useful technique in term extraction (Frantzi et al., 2000), a domain closely related to named-entity recognition. Here, terms are extracted with a predefined part-of-speech structure, e.g. an adjective-noun combination. In (Nenadić et al., 2002), methods are discussed to extract information from natural language texts with the use of both part of speech tags and hyponym patterns.

As referred to in the introduction, Ravichandran and Hovy (2002) present a method to identify surface text patterns using a web search engine. They extract patterns expressing functional relations in a factoid question answering setting. Selection of the extracted patterns is based on the precision of the patterns. For example, if the pattern 'was born in' is identified as a pattern for the pair ('Mozart', 'Salzburg'), they compute precision as the number of excerpts containing 'Mozart was born in Salzburg' divided by the number of excerpts with 'Mozart was born in'.

Information extraction and ontologies creation are two closely related fields. For reliable information extraction, we need background information, e.g. an ontology. On the other hand, we need information extraction to generate broad and highly usable ontologies. An overview on ontology learning from text can be found in (Buitelaar et al., 2005).

Early work (Hearst, 1998), describes the extraction of text patterns expressing WordNet-relations (such as hyponym relations) from some corpus. This work focusses merely on the identification of such text patterns (i.e. phrases containing both instances of some related pair). Patterns found by multiple pairs are suggested to be usable patterns.

KnowItAll is a hybrid named-entity extraction system (Etzioni et al., 2005) that finds lists of instances of some class from the web using a search engine. It combines Hearst patterns and learned

patterns for instances of some class to identify and extract named-entities. Moreover, it uses adaptive wrapper algorithms (Crescenzi and Mecca, 2004) to extract information from html markup such as tables.

Cimiano and Staab descibe a method to use a search engine to verify a hypothesis relation (2004). For example, if we are interested in the 'is a' or hyponym relation and we have a candidate instance pair ('river', 'Nile') for this relation, we can use a search engine to query phrases expressing this relation (e.g. 'rivers such as the Nile'). The number of hits to such queries can then be used as a measure to determine the validity of the hypothesis.

In (Geleijnse and Korst, 2005), a method is described to populate an ontology with the use of queried text patterns. The algorithm presented extracts instances from search results after having submitted a combination of an instance and a pattern as a query to a search engine. The extracted instances from the retrieved excerpts can thereafter be used to formulate new queries – and thus identify and extract other instances.

4   The algorithm

We present an algorithm to learn surface text patterns for relations. We use Google™ to retrieve such patterns.

The algorithm makes use of a training set $T_{\mathcal{R}}$ of instance-pairs that are $\mathcal{R}$-related. This training set should be chosen such the instance-pairs are typical for relation $\mathcal{R}$.

We first discover how relation $\mathcal{R}$ is expressed in natural language texts on the web (Section 4.1). In Section 4.2 we address the problem of selecting *effective* patterns from the total set of patterns found.

4.1   Identifying relation patterns

We first generate a list of surface text patterns with the use of the following algorithm. For evaluation purposes, we also compute the frequency of each pattern found.

- **Step 1:** Formulate queries using an instance-pair $(x, y) \in T_{\mathcal{R}}$. Since we are interested in phrases within sentences rather than in keywords or expressions in telegram style that often appear in titles of webpages, we use the `allintext:` option. This gives us only search results with the queried expression in

the bodies of the documents rather than in the titles. We query both `allintext:"` x `*` y `"` and `allintext:"` y `*` x `"`. The `*` is a regular expression operator accepted by Google. It is a placeholder for zero or more words.

- **Step 2:** Send the queries to Google and collect the excerpts of the at most 1,000 pages it returns for each query.

- **Step 3:** Extract all phrases matching the queried expressions and replace both x and y by the names of their classes.

- **Step 4:** Remove all phrases that are not within one sentence.

- **Step 5:** Normalize all phrases by removing all mark-up that is ignored by Google. Since Google is case-insensitive and ignores punctuation, double spaces and the like, we translate all phrases found to a normal form: the simplest expression that we can query that leads to the document retrieved.

- **Step 6:** Update the frequencies of all normalized phrases found.

- **Step 7:** Repeat the procedure for any unqueried pair $(x', y') \in T_{\mathcal{R}}$.

We now have generated a list with relation patterns and their frequencies within the retrieved Google excerpts.

4.2   Selecting relation patterns

From the list of relation patterns found, we are interested in the most effective ones.

We are not only interested in the most precise ones. For example, the retrieved pattern "född 30 mars 1853 i" proved to a 100% precise pattern expressing the relation between a person ('Vincent van Gogh') and his place of birth ('Zundert'). Clearly, this rare phrase is unsuited to mine instance-pairs of this relation in general. On the other hand, high frequency of some pattern is no guarantee for effectiveness either. The frequently occurring pattern "was born in London" (found when querying for `Thomas Bayes * England`) is well-suited to be used to find London-born persons, but in general the pattern is unsuited – since too narrow – to express the relation between a person and his or her country of origin.

Taking these observations into account, we formulate three criteria for selecting effective relation patterns.

1. The patterns should *frequently* occur on the web, to increase the probability of getting any results when querying the pattern in combination with an instance.

2. The pattern should be *precise*. When we query a pattern in combination with an instance in I_q, we want to have many search results containing instances from c_a.

3. If relation $\mathcal{R}$ is not functional, the pattern should be *wide-spread*, i.e. among the search results when querying a combination of the pattern and an instance in I_q there must be as many distinct $\mathcal{R}$-related instances from c_a as possible.

To measure these criteria, we use the following scoring functions for relation patterns s.

1. $f_{\text{freq}}(s) =$ "number of occurrences of s in the excerpts as found by the algorithm described in the previous subsection"

2. $f_{\text{prec}}(s) = \frac{\sum_{x \in I'_q} P(s,x)}{|I'_q|}$, where

   for instances $x \in I'_q$, $I'_q \subseteq I_q$, we calculate $P(s,x)$ as follows.
   $P(s,x) = \frac{F_I(s,x)}{F_O(s,x)}$
   and
   $F_I(s,x) =$ the number of Google excerpts after querying s in combination with x containing instances of c_a.
   $F_O(s,x) =$ the total number of excerpts found (at most 1,000).

3. $f_{\text{spr}}(s) = \sum_{x \in I'_q} B(s,x)$, where

   $B(s,x) =$ the number of *distinct* instances of class c_a found after querying pattern s in combination with x.

The larger we choose the testset, the subset I'_q of I_q, the more reliable the measures for precision and spreading. However, the number of Google queries increases with the number of patterns found for each instance we add to I'_q.

We finally calculate the score of the patterns by multiplying the individual scores:

$$\text{score}(s) = f_{\text{freq}}(s) \cdot f_{\text{prec}}(s) \cdot f_{\text{spr}}(s)$$

For efficiency reasons, we only compute the scores of the patterns with the highest frequencies.

The problem remains how to recognize a (possible multi-word) instance in the Google excerpts. For an ontology alignment setting – where the sets I_a and I_q are not to be expanded – these problems are trivial: we determine whether $t \in I_a$ is accompanied by the queried expression. For a setting where the instances of c_a are not all known (e.g. it is not likely that we have a complete list of all books written in the world), we solve this problem in two stages. First we identify rules per class to extract candidate instances. Thereafter we use an additional Google query to verify if a candidate is indeed an instance of class c_a.

Identifying a candidate instance

The identification of multi-word terms is an issue of research on its own. However, in this setting we can allow ourselves to use less elaborate techniques to identify candidate instances. We can do so, since we additionally perform a check on each extracted term. So, per class we create rules to identify candidate instances with a focus on high recall. In our current experiments we thus use very simple term recognition rules, based on regular expressions. For example, we identify a candidate instance of class 'person' if the queried expression is accompanied by two or three capitalized words.

Identifying an instance-class relation

We are interested in the question whether some extracted term t is an instance of class c_a. For example, given the term 'The Godfather', does this term belong to the class 'movie'? The instance-class relation can be viewed of as a hyponym relation. We therefore verify the hypothesis of t being an instance of c_a by Googling hyponym relation patterns. We use a fixed set H of common patterns expressing the hyponym relation (Hearst, 1992; Cimiano and Staab, 2004), see Table 1. For the class names, we use plurals.

We use these patterns in the following acceptance function

$$\text{accept}_{c_q}(t) := (\sum_{p \in H} h(p, c_q, t) \geq n),$$

`"`c_q `including` t `and"`
`"`c_q `for example` t `and"`
`"`c_q `like` t `and"`
`"`c_q `such as` t `and"`

Table 1: Hearst patterns for instance-class relation.

where $h(p, c_q, t)$ is the number of Google hits for query with pattern p combined with term t and the plural form of the class name c_q. The threshold n has to be chosen beforehand. We can do so, by calculating the sum of Google hits for queries with known instances of the class. Based on these figures, a threshold can be chosen e.g. the minimum of these sums.

Note that term t is both preceded and followed by a fixed phrase in the queries. We do so, to guarantee that t is indeed the full term we are interested in. For example, if we had extracted the term 'Los' instead of 'Los Angeles' as a Californian City, we would falsely identify 'Los' as a Californian City, when we do not let 'Los' follow by the fixed expression `and`. The number of Google hits for some expression x is at least the number of Google hits when querying the same expression followed by some expression y.

If we identify a term t as being an instance of class c_a, we can add this term to the set I_a. However, we cannot relate t to an instance in I_q, since the pattern used to find t has not proven to be effective yet (e.g. the pattern could express a different relation between one of the instance-pairs in the training set).

We reduce the amount of Google queries by using a list of terms found that do not belong to c_a. Terms that occur multiple times in the excerpts can then be checked only once. Moreover, we use the `OR`-clause to combine the individual queries into one. We then check if the number of hits to this query exceeds the threshold. The amount of Google queries in this phase thus equals the amount of distinct terms extracted.

5 The use of surface text patterns in information extraction

Having a method to identify relation patterns, we now focus on utilizing these patterns in information extraction from texts found by a search engine. We use an ontology to represent the information extracted.

Suppose we have an ontology O with classes $(c_1, c_2, ...)$ and corresponding instance sets $(I_1, I_2, ..)$. On these classes, relations $\mathcal{R}_{(i,j)}$[1] are defined, with i and j the index number of the classes. The non-empty sets $T_{(i,j)}$ contain the training set of instance-pairs of the relations $\mathcal{R}_{(i,j)}$.

Per instance, we maintain a list of expressions that already have been used as a query. Initially, these are empty.

The first step of the algorithm is to learn surface text patterns for each relation in O.

The following steps of the algorithm are performed until either some stop criterion is reached, or no more new instances and instance-pairs can be found.

- **Step 1:** Select a relation $\mathcal{R}_{(i,j)}$, and an instance v from either I_i or I_j such that there exists at least one pattern expressing $\mathcal{R}_{(i,j)}$ we have not yet queried in combination with v.

- **Step 2:** Construct queries using the patterns with v and send these queries to Google.

- **Step 3:** Extract instances from the excerpts.

- **Step 4:** Add the newly found instances to the corresponding instance set and add the instance-pairs found (thus with v) to $T_{(i,j)}$.

- **Step 5:** If there exists an instance that we can use to formulate new queries, then repeat the procedure.
  Else, learn new patterns using the extracted instance-pairs and then repeat the procedure.

Note that instances of class c_x learned using the algorithm applied on relation $\mathcal{R}_{(x,y)}$ can be used as input for the algorithm applied to some relation $\mathcal{R}_{(x,z)}$ to populate the sets I_z and $T_{(x,z)}$.

6 Experiments

In this section, we discuss two experiments that we have conducted. The first experiment involves the identification of effective hyponym patterns. The second experiment is an illustration of the application of learned surface text patterns in information extraction.

---

[1]Assuming one relation per pair of classes. We can use another index k in $\mathcal{R}_{(i,j,k)}$ to distinct multiple relations between c_i and c_j.

6.1 Learning effective hyponym patterns

We are interested whether the effective surface text patterns are indeed intuitive formulations of some relation $\mathcal{R}$. As a test-case, we compute the most effective patterns for the hyponym relation using a test set with names of all countries.

Our experiment was set up as follows. We collected the complete list of countries in the world from the CIA World Factbook[2]. Let I_q be this set of countries, and let I_a be the set { 'countries', 'country' }. The set $T_{\mathcal{R}}$ consists of all pairs $(a,$ 'countries') and $(a,$ 'country') , for $a \in I_a$. We apply the surface text pattern learning algorithm on this set $T_{\mathcal{R}}$.

The algorithm identified almost 40,000 patterns. We computed f_{spr} and f_{prec} for the 1,000 most frequently found patterns. In table 2, we give the 25 most effective patterns found by the algorithm. We consider the patterns in boldface true hyponym patterns. Focussing on these patterns, we observe two groups: 'is a' and Hearst-like patterns.

pattern	freq	prec	spr
(countries) like	645	0.66	134
(countries) such as	537	0.54	126
is a small (country)	142	0.69	110
(country) code for	342	0.36	84
(country) map of	345	0.34	78
(countries) including	430	0.21	93
is the only (country)	138	0.55	102
is a (country)	339	0.22	99
(country) flag of	251	0.63	46
and other (countries)	279	0.34	72
and neighboring (countries)	164	0.43	92
(country) name republic of	83	0.93	76
(country) book of	59	0.77	118
is a poor (country)	63	0.73	106
is the first (country)	53	0.70	112
(countries) except	146	0.37	76
(country) code for calling	157	0.95	26
is an independent (country)	62	0.55	114
and surrounding (countries)	84	0.40	107
is one of the poorest (countries)	61	0.75	78
and several other (countries)	65	0.59	90
among other (countries)	84	0.38	97
is a sovereign (country)	48	0.69	89
or any other (countries)	87	0.58	58
(countries) namely	58	0.44	109

Table 2: Learned hyponym patterns and their scores.

The Hearst-patterns 'like' and 'such as' show to be the most effective. This observation is useful, when we want to minimize the amount of queries for hyponym patterns.

Expressions of properties that hold for each

country and only for countries, for example the existence of a country code for dialing, are not trivially identified manually but are useful and reliable patterns.

The combination of 'is a', 'is an' or 'is the' with an adjective is a common pattern, occurring 2,400 times in the list. In future work, we plan to identify such adjectives in Google excerpts using a Part of Speech tagger (Brill, 1992).

6.2 Applying learned patterns in information extraction

The Text Retrieval Conference (TREC) question answering track in 2004 contains list question, for example 'Who are Nirvana's band members?' (Voorhees, 2004). We illustrate the use of our ontology population algorithm in the context of such list-question answering with a small case-study. Note that we do not consider the processing of the question itself in this research.

Inspired by one of the questions ('What countries is Burger King located in?'), we are interested in populating an ontology with restaurants and the countries in which they operate. We identify the classes 'country' and 'restaurant' and the relation 'located in' between the classes.

We hand the algorithm the instances of 'country', as well as two instances of 'restaurant': 'McDonald's' and 'KFC'. Moreover, we add three instance-pairs of the relation to the algorithm. We use these pairs and a subset I'_{country} of size eight to compute a ranked list of the patterns. We extract terms consisting of one up to four capitalized words. In this test we set the threshold for the number of Google results for the queries with the extracted terms to 50. After a small test with names of international restaurant branches, this seemed an appropriate threshold.

The algorithm learned, besides a ranked list of 170 surface text patterns (Table 3), a list of 54 instances of restaurant (Table 4). Among these instances are indeed the names of large international chains, Burger King being one of them. Less expected are the names of geographic locations and names of famous cuisines such as 'Chinese' and 'French'. The last category of false instances found that have not be filtered out, are a number of very common words (e.g. 'It' and 'There').

We populate the ontology with relations found between Burger King and instances from country using the 20 most effective patterns.

pattern	prec	spr	freq
c_a restaurants of c_q	0.24	15	21
c_a restaurants in c_q	0.07	19	9
c_a hamburger chain that occupies villages throughout modern day c_q	1.0	1	7
c_a restaurant in c_q	0.06	16	6
c_a restaurants in the c_q	0.13	16	2
c_a hamburger restaurant in southern c_q	1.0	1	4

Table 3: Top learned patterns for the restaurant-country (c_a - c_q) relation.

Chinese	Bank	Outback Steakhouse
Denny's	Pizza Hut	Kentucky Fried Chicken
Subway	Taco Bell	Continental
Holywood	Wendy's	Long John Silver's
HOTEL OR	This	**Burger King**
Japanese	West	Keg Steakhouse
You	BP	Outback
World	Brazil	San Francisco
Leo	Victoria	New York
These	Lyons	Starbucks
FELIX	Roy	California Pizza Kitchen
Marks	Cities	Emperor
Friendly	Harvest	Friday
New York	Vienna	Montana
Louis XV	Greens	Red Lobster
Good	It	There
That	Mark	Dunkin Donuts
Italia	French	Tim Hortons

Table 4: Learned instances for restaurant.

The algorithm returned 69 instance-pairs with countries related to 'Burger King'. On the Burger King website[3] a list of the 65 countries can be found in which the hamburger chain operates. Of these 65 countries, we identified 55. This implies that our results have a precision of $\frac{55}{69} = 80\%$ and recall of $\frac{55}{65} = 85\%$. Many of the falsely related countries – mostly in eastern Europe – are locations where Burger King is said to have plans to expand its empire.

7 Conclusions

We have presented a novel approach to identify useful surface text patterns for information extraction using an internet search engine. We argued that the selection of patterns has to be based on effectiveness: a pattern has to occur frequently, it has to be precise and has to be wide-spread if it represents a non-functional relation.

These criteria are combined in a scoring function which we use to select the most effective patterns.

---

[3]http://www.whopper.com

The method presented can be used for arbitrary relations, thus also relations that link an instance to multiple other instances. These patterns can be used in information extraction. We combine patterns with an instance and offer such an expression as a query to a search engine. From the excerpts retrieved, we extract instances and simultaneously instance-pairs.

Learning surface text patterns is efficient with respect to the number of queries if we know all instances of the classes concerned. The first part of the algorithm is linear to the size of the training set. Furthermore, we select the n most frequent patterns and perform $|I'_q| \cdot n$ queries to compute the score of these n patterns.

However, for a setting where I'_a is incomplete, we have to perform a check for each unique term identified as a candidate instance in the excerpts found by the $|I'_q| \cdot n$ queries. The number of queries, one for each extracted unique candidate instance, thus fully depends on the rules that are used to identify a candidate instance.

We apply the learned patterns in an ontology population algorithm. We combine the learned high quality relation patterns with an instance in a query. In this way we can perform a range of effective queries to find instances of some class and simultaneously find instance-pairs of the relation.

A first experiment, the identification of hyponym patterns, showed that the patterns identified indeed intuitively reflect the relation considered. Moreover, we have generated a ranked list of hyponym patterns. The experiment with the restaurant ontology illustrated that a small training set suffices to learn effective patterns and populate an ontology with good precision and recall. The algorithm performs well with respect to recall of the instances found: many big international restaurant branches were found. The identification of the instances however is open to improvement, since the additional check does not filter out all falsely identified candidate instances.

8 Future work

Currently we check whether an extracted term is indeed an instance of some class by querying hyponym patterns. However, if we find two instances related by some surface text pattern, we always accept these instances as instance pair. Thus, if we both find 'Mozart was born in Germany' and 'Mozart was born in Austria', both extracted

instance-pairs are added to our ontology. We thus need some post-processing to remove falsely found instance-pairs. When we know that a relation is functional, we can select the most frequently occurring instance-pair.

Moreover, the process of identifying an instance in a text needs further research especially since the method to identify instance-class relations by querying hyponym patterns is not flawless.

The challenge thus lies in the area of improving the precision of the output of the ontology population algorithm. With additional filtering techniques and more elaborated identification techniques we expect to be able to improve the precision of the output. We plan to research check functions based on enumerations of candidate instances with known instances of the class. For example, the enumeration 'KFC, Chinese and Mc-Donald's' is not found by Google, where 'KFC, Burger King and McDonald's' gives 31 hits.

Our experiment with the extraction of hyponym patterns, suggests a ranking of Hearst-patterns based on the effectiveness. Knowledge on the effectiveness of each of the Hearst-patterns can be utilized to minimize the amount of queries.

Finally we will investigate ways to compare our methods with other systems in a TREC like setting with the web as a corpus.

Acknowledgments

References

E. Agichtein and L. Gravano. 2000. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the Fifth ACM International Conference on Digital Libraries*.

E. Brill. 1992. A simple rule-based part-of-speech tagger. In *Proceedings of the third Conference on Applied Natural Language Processing (ANLP'92)*, pages 152–155, Trento, Italy.

S. Brin. 1998. Extracting patterns and relations from the world wide web. In *WebDB Workshop at sixth International Conference on Extending Database Technology (EDBT'98)*.

P. Buitelaar, P. Cimiano, and B. Magnini, editors. 2005. *Ontology Learning from Text: Methods, Evaluation and Applications*, volume 123 of *Frontiers in Artificial Intelligence and Applications*. IOS Press.

N. A. Chinchor, editor. 1998. *Proceedings of the Seventh Message Understanding Conference (MUC-7)*. Morgan Kaufmann, Fairfax, Virginia.

P. Cimiano and S. Staab. 2004. Learning by googling. *SIGKDD Explorations Newsletter*, 6(2):24–33.

M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. 2000. Learning to construct knowledge bases from the World Wide Web. *Artificial Intelligence*, 118:69–113.

V. Crescenzi and G. Mecca. 2004. Automatic information extraction from large websites. *Journal of the ACM*, 51(5):731–779.

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

K. Frantzi, S. Ananiado, and H. Mima. 2000. Automatic recognition of multi-word terms: the c-value/nc-value method. *International Journal on Digital Libraries*, 3:115–130.

G. Geleijnse and J. Korst. 2005. Automatic ontology population by googling. In *Proceedings of the Seventeenth Belgium-Netherlands Conference on Artificial Intelligence (BNAIC 2005)*, pages 120 – 126, Brussels, Belgium.

M. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics*, pages 539–545, Morristown, NJ, USA.

M. Hearst. 1998. Automated discovery of wordnet relations. In Christiane Fellbaum, editor, *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA.

G. Nenadić, I. Spasić, and S. Ananiadou. 2002. Automatic discovery of term similarities using pattern mining. In *Proceedings of the second international workshop on Computational Terminology (CompuTerm'02)*, Taipei, Taiwan.

D. Ravichandran and E. Hovy. 2002. Learning surface text patterns for a question answering system. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL 2002)*, pages 41–47, Philadelphia, PA.

E. Voorhees. 2004. Overview of the trec 2004 question answering track. In *Proceedings of the 13th Text Retrieval Conference (TREC 2004)*, Gaithersburg, Maryland.

G. Zhou and J. Su. 2002. Named entity recognition using an hmm-based chunk tagger. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL 2002)*, pages 473 – 480, Philadelphia, PA.