Combining Linguistic and Statistical Analysis to Extract Relations from Web Documents

Fabian M. Suchanek
Max-Planck-Institute for
Computer Science
Saarbrücken/Germany
suchanek@ mpii.mpg.de

Georgiana Ifrim
Max-Planck-Institute for
Computer Science
Saarbrücken/Germany
ifrim@ mpii.mpg.de

Gerhard Weikum
Max-Planck-Institute for
Computer Science
Saarbrücken/Germany
weikum@ mpii.mpg.de

ABSTRACT

The World Wide Web provides a nearly endless source of knowledge, which is mostly given in natural language. A first step towards exploiting this data automatically could be to extract pairs of a given semantic relation from text documents – for example all pairs of a person and her birthdate. One strategy for this task is to find text patterns that express the semantic relation, to generalize these patterns, and to apply them to a corpus to find new pairs. In this paper, we show that this approach profits significantly when deep linguistic structures are used instead of surface text patterns. We demonstrate how linguistic structures can be represented for machine learning, and we provide a theoretical analysis of the pattern matching approach. We show the practical relevance of our approach by extensive experiments with our prototype system LEILA.

Categories and Subject Descriptors

I.2.7 [**Artificial Intelligence**]: Natural Language Processing - text analysis; I.2.6 [**Artificial Intelligence**]: Learning - knowledge acquisition

General Terms

Algorithms, Design, Experimentation, Theory

1. INTRODUCTION

1.1 Motivation

Many data mining tasks such as classification, ranking, recommendation, or data cleaning could be boosted by explicit formalized world knowledge. Unfortunately, the manual construction and maintenance of such knowledge bases is a limiting factor in our modern world of "exploding information". Hence it seems tempting to exploit the World Wide Web and other poorly structured information sources for automatically acquiring ontological knowledge. In this context, a first step could be to extract instances of a given target relation from a given Web page corpus. For example, one

might be interested in extracting all pairs of a person and her birth date (the `birthdate`-relation), all pairs of a company and the city of its headquarters (the `headquarters`-relation) or all pairs of an entity and the class it belongs to (the `instanceOf`-relation).

The most promising techniques to extract information from unstructured text seem to be natural language processing (NLP) techniques. Most approaches, however, have limited the NLP part to part-of-speech tagging. This paper demonstrates that information extraction can profit significantly from deep natural language processing. It shows how deep syntactic structures can be represented suitably and it provides a statistical analysis of the pattern matching approach.

1.2 Related Work

There are numerous Information Extraction (IE) approaches. Some focus on unary relations (e.g. on extracting all `cities` from a given text [13, 7]). In this paper we pursue the more general binary relations. Some systems are designed to discover new binary relations [21]. However, in our setting, the target relation is given. Some systems are restricted to learning the `instanceOf`-relation [11, 4]. By contrast, we are interested in extracting arbitrary relations (including `instanceOf`). Whereas there are systems that require human input for the IE process [24], our work aims at a completely automated system. There exist systems that can extract information efficiently from formatted data [15, 14]. However, since a large part of the Web consists of natural language text, we consider in this paper only systems that accept unstructured corpora. As initial input, some systems require a hand-tagged corpus [17, 31], manually assembled text patterns [34] or hand-chosen templates [32]. Since manually tagged input amounts to huge human effort, we consider here only systems that do not have this constraint. Some systems do not work on a closed corpus, but make use of the full Web for the IE process [12, 9]. Despite the more powerful setting, these systems use extraction techniques similar to the other approaches. In order to study these extraction techniques in a controlled environment, we restrict ourselves to corpus-based systems for this paper.

One school of extraction techniques concentrates on detecting the boundaries of interesting entities in the text [7, 13, 35]. This usually goes along with the restriction to unary target relations. Other approaches make use of the context in which an entity appears [10, 5]. This school is restricted to the `instanceOf`-relation. The only group that can learn arbitrary binary relations is the group of pattern matching systems. The huge majority of them [12, 1, 23, 3, 28, 33] uses only a shallow linguistic analysis of the cor-

pus. Consequently, most of them are extremely volatile to small variations in the patterns (see the conclusion of [23] for an example). Furthermore, these approaches cannot benefit from advanced linguistic techniques such as anaphora resolution. The few approaches that do use deep NLP [6, 27] consider only the shortest path in the dependency graph as a feature. Thus, these systems cannot deal with the difference between "A dog is a mammal" (which expresses the subConcept-relation) and "This dog is a nag" (which does not). None of the pattern matching approaches provides an analysis of the influence of false positive patterns.

1.3   Link Grammars

There exist different approaches for parsing natural language sentences. They range from simple part-of-speech tagging to context-free grammars and more advanced techniques such as Lexical Functional Grammars, Head-Driven Phrase Structure Grammars or stochastic approaches. For our implementation, we chose the Link Grammar Parser [26]. It is based on a context-free grammar and hence it is simpler to handle than the advanced parsing techniques. At the same time, it provides a much deeper semantic structure than the standard context-free parsers. Figure 1 shows a linguistic structure produced by the Link Parser (a *linkage*). A linkage is a connected planar undirected graph, the
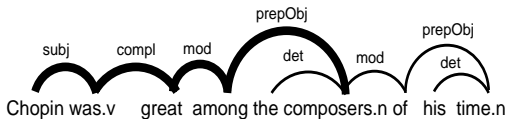

Figure 1: A simple linkage

nodes of which are the words of the sentence. The edges (the *links*) are labeled with *connectors*. For example, the connector **subj** marks the link between the subject and the verb of the sentence. The linkage must fulfill certain linguistic constraints. These are given by a *link grammar*, which specifies which word may be linked by which connector to preceding and following words. The parser also assigns *part-of-speech tags*. For example, in Figure 1, the suffix ".n" identifies "composers" as a noun.

We say that a linkage *expresses* a relation r, if the underlying sentence implies that a pair of entities is in r. Note that the deep grammatical analysis of the sentence would allow us to define the meaning of the sentence in a theoretically well-founded way [22]. For this paper, however, we limit ourselves to an intuitive understanding of the notion of meaning. The problem of the corpus containing sentences that are not true is outside the scope of this paper.

We define a *pattern* as a linkage in which two words have been replaced by placeholders. Figure 2 shows a sample pattern with the placeholders "X" and "Y". We call the
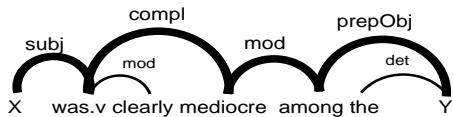

Figure 2: A simple pattern

(unique) shortest path from one placeholder to the other the *bridge*, marked in bold in Figure 2. A pattern *matches* a linkage if the bridge of the pattern appears in the linkage, although nouns and adjectives are allowed to differ. For example, the pattern in Figure 2 matches the linkage in Figure 1, because the bridge of the pattern occurs in the linkage, apart from a substitution of "great" by "mediocre".

If a pattern matches a linkage, we say that the pattern *produces* the pair of words that the linkage contains in the posi-

tion of the placeholders. In our example, the pair "Chopin" / "composers" is produced.

2.   SYSTEM MODEL

2.1   Algorithm

As a definition of the target relation, our algorithm requires a function that decides into which of the following categories a pair of words falls:

- An **example** for the target relation. For instance, for the birthdate-relation, the examples can be given by a list of persons with their birth dates.
- A **counterexample**. For the birthdate-relation, the counterexamples can be deduced from the examples (e.g. if "Chopin" / "1810" is an example, then "Chopin" / "2000" must be a counterexample).
- A **candidate**. For birthdate, the candidates would be all pairs of a proper name and a date that are not an example or a counterexample (e.g. if "Mozart" is not in the examples, then "Mozart" / "2000" is a candidate).
- None of the above.

The corpus should be a sequence of natural language sentences. These sentences are parsed, producing a deep *grammatical structure* for each of them. In principle, our algorithm does not depend on a specific parsing technique. For example, the parse-trees produced by a context-free grammar can serve as grammatical structures. Here, we use linkages. The core algorithm proceeds in three phases:

1. In the *Discovery Phase*, it seeks linkages in which an example pair appears. It replaces the two words by placeholders, thus producing a pattern. These patterns are collected as *positive patterns*. Then, the algorithm runs through the sentences again and finds all linkages that match a positive pattern, but produce a counterexample. The corresponding patterns are collected as *negative patterns*[1].

2. In the *Training Phase*, statistical learning is applied to learn the concept of positive patterns. The result of this process is a classifier for patterns.

3. In the *Testing Phase*, the algorithm considers again all sentences in the corpus. For each linkage, it generates all possible patterns by replacing two words by placeholders. If the two words form a candidate and the pattern is classified as positive, the produced pair is proposed as a new element of the target relation (an *output pair*).

Although usually the Discovery Phase and the Testing Phase are run on the same corpus, it is also possible to run them on two distinct corpora.

2.2   Robust Learning

The central task of the Discovery Phase is determining patterns that express the target relation. Since the linguistic meaning of the patterns is not apparent to the system, it relies on the following **hypothesis**: Whenever an example pair appears in a sentence, the linkage and the corresponding pattern express the target relation. This hypothesis may fail if a sentence contains an example pair merely by chance, i.e. without expressing the target relation. In this case we would use the pattern as a positive sample for the generalization process, although it is a negative one. Analogously, a pattern that does express the target relation may occasionally produce counterexamples. In this case, the pattern is used as a negative sample in the generalization process. We

---

[1]Note that different patterns can match the same linkage.

call these patterns *false samples*. The problem of false samples is intrinsic for pattern matching approaches in general. However, false samples do not question the effectiveness of our approach.

This is because virtually any learning algorithm can deal with a limited number of false samples. For Support Vector Machines (SVM), the effect of false samples has been analyzed thoroughly in [8]. In general, an SVM is highly tolerant to noise. There are also detailed theoretical studies [2] on how the proportion of false samples influences a PAC-learner. In essence, the number of required samples increases, but the classification is still learnable. It is also possible to understand the concept of positive patterns as a probabilistic concept [19]. In this setting, the pattern is not classified as either positive or negative, but it may produce pairs of the target relation with a certain fixed probability. The task of the learner is to learn the function from the pattern to its probability. [25] shows that probabilistic concepts can be learned and gives bounds on the number of required samples. The following subsection considers a particularly simple class of learners, the k-Nearest-Neighbor-classifiers.

2.2.1 k-Nearest-Neighbor Classifiers

A k-Nearest-Neighbors (kNN) classifier requires a distance function on patterns. We consider a simple variant of an adaptive kNN classifier: In the Discovery Phase, a newly discovered pattern becomes a *prototype* for a whole class of new patterns. Whenever another pattern is discovered, we check whether its distance to an existing prototype is below some threshold θ. We say that the pattern *falls on* the prototype[2]. If the new pattern does not fall on an existing prototype, it becomes a prototype on its own. After the Discovery Phase, we label a prototype as *positive* if the majority of the patterns that fell on it were positive, as *negative* else.

In the Testing Phase, we find for each test pattern its closest prototype. If there is no prototype within the distance θ, the pattern is classified as negative. If it falls on a prototype p, the pattern is classified as positive if p has a positive label and as negative else. We are interested in the probability that a test pattern is classified as positive, although the produced pair is not in the target relation.

In the Testing Phase, each possible pattern is generated for each sentence in the corpus (this will be a number of patterns quadratic in the number of nouns in the sentence). We model the sequence of all these patterns as a sequence of N random events. Each pattern produces a pair of words with its underlying sentence. This pair can either be an example, a counterexample or a candidate[3]. We model these events by Bernoulli random variables $EX, CE, CAND$, captured by a multinomial distribution: $EX = 1$ iff the pair is an example, $CE = 1$ iff the pair is a counterexample, $CAND = 1 - EX - CE = 1$ iff the pair is a candidate. For each prototype p, we introduce a Bernoulli random variable F_p, such that $F_p = 1$ with probability f_p iff a generated pattern falls on p. Note that this model also applies to the Discovery Phase.

We first concentrate on the Discovery Phase. We are interested in the probability that a given prototype p gets a positive label, although it does not express the target rela-

---

[2]If θ is chosen sufficiently small, all patterns falling on p share their essential linguistic properties. Hence we assume that they all have the same probability of producing examples or counterexamples.

[3]For simplification, we assume that the 4th class of word pairs (see section 2.1) does not appear. If it does, it will only improve the bound given here.

tion. We define the *quality* of p as the relative probability of patterns falling on p to produce examples:

$$q_p = \frac{P(EX|F_p)}{P(EX|F_p) + P(CE|F_p)}$$

Since p does not express the target relation, $q_p < \frac{1}{2}$. The *allotment* of p is the share of examples and counterexamples produced by patterns falling on p: $a_p = P(EX|F_p) + P(CE|F_p)$. The better the examples and counterexamples are chosen, the more likely it is that patterns falling on p produce examples or counterexamples (instead of candidates) and the larger a_p will be. Let $\#EX_p$ stand for the number of examples and $\#CE_p$ for the number of counterexamples produced by patterns falling on p in the Discovery Phase. We are interested in the probability of p getting a positive label, namely $P(\#EX_p > \#CE_p)$, given that $q_p < \frac{1}{2}$. Using Chernoff-Hoeffding bounds, we prove[29] that

$$P(\#EX_p > \#CE_p) \leq 2e^{-N\frac{1}{2}(a_p f_p)^2} + 2e^{-(a_p f_p N + 2)(\frac{1}{2} - q_p)^2} .$$

Now we turn to the Testing Phase. We are interested in the probability that an incorrect output pair is produced by a pattern falling on p. For this to happen, a test pattern must fall on p, it must produce a candidate and p must be wrongly labeled as positive. Combined, this yields

$$
\begin{aligned}
&P(CAND \cap F_p) \cdot P(\#EX_p > \#CE_p) \\
=\ &(1 - a_p) \cdot f_p \cdot P(\#EX_p > \#CE_p) \\
\leq\ &2(1 - a_p) \cdot f_p \cdot (e^{-N\frac{1}{2}(a_p f_p)^2} + e^{-(a_p f_p N + 2)(\frac{1}{2} - q_p)^2}) .
\end{aligned}
$$

This estimation shows that a larger allotment a_p (i.e. a good choice of examples and counterexamples) decreases the probability of wrongly classifying a candidate pair. Furthermore, the estimation mirrors the intuition that either many patterns fall on p in the Discovery Phase (f_p large) and then p is unlikely to have a false label, or few patterns fall on p (f_p small) and then the probability of p classifying a test pattern is small. As the number of sentences (and hence the number of generated patterns N) increases, the bound converges to zero.

2.3 Feature Model

This section discusses how patterns can be represented and generalized using machine learning. The most important component of a pattern is its bridge. In the Discovery Phase, we collect the bridges of the patterns in a list. Each bridge is given an identification number, the *bridge id*. Two bridges are given the same bridge id if they differ only in their nouns or adjectives (as discussed in section 1.3). Furthermore, positive patterns are given the *label* +1 and negative patterns −1. The *context* of a word in a linkage is the set of all its links together with their direction in the sentence (left or right) and their target words. For example, the context of "composers" in Figure 1 is the set of triples {(det, left, "the"), (prepObj, left, "among"), (mod, right, "of")}. Each word is assigned a set of *types*. We distinguish nouns, adjectives, prepositions, verbs, numbers, dates, names, person names, company names and abbreviations. The parser already assigns the grammatical types by its part-of-speech tagging. We assign the other types by regular expression matching. For example, any word matching "[A-Z][a-z]+ Inc" is given the type company. To accommodate the role of stopwords in understanding a sentence, we make each stopword a type of its own. We represent a pattern by a quadruple of its bridge id, the context of the first placeholder, the context of the second placeholder, and its label. For example, supposing that the bridge id of the pattern in Figure 2 is 42 and supposing that the pattern is

positive, we represent the pattern as

```
(42, {(subj,right,"was")}, {(det, left, "the"),
(prepObj, left, "among"), (ofComp, right, "of")}, +1)
```

To show that our approach does not depend on a specific learning algorithm, we implemented two machine learning algorithms: The adaptive kNN classifier discussed in 2.2.1 and an SVM classifier.

2.3.1 kNN

For the adaptive kNN, we need a similarity function on patterns. By $x \sim y$ we denote the auxiliary function

$$x \sim y \;=\; \begin{cases} 1 & \text{if } x = y \\ 0 & \text{else} \end{cases}$$

Let $\tau(w)$ be the set of types of a word w. The similarity of two words is the overlap of their type sets:

$$sim(w_1, w_2) = \frac{|\tau(w_1) \cap \tau(w_2)|}{|\tau(w_1) \cup \tau(w_2)|}$$

The similarity of two contexts C_1, C_2 is computed by comparing each triple in C_1 to all triples in C_2, where each triple contains a connector, a direction and a word:

$$sim(C_1, C_2) =$$

$$\sum_{\substack{(con_1, dir_1, w_1) \in C_1 \\ (con_2, dir_2, w_2) \in C_2}} \frac{\alpha_1 (con_1 \sim con_2) + \alpha_2 (dir_1 \sim dir_2) + \alpha_3 sim(w_1, w_2)}{|C_1| \cdot |C_2|}$$

Here, α_1, α_2, α_3 are weighting factors that sum up to 1. We chose $\alpha_1 = 0.4$, $\alpha_2 = 0.2$, $\alpha_3 = 0.4$. Two patterns have a similarity of zero if they have different bridge ids. Else, their similarity is the averaged similarity of the contexts of the first and second placeholder, respectively:

$$sim((b_1, C_{11}, C_{12}, l_1), (b_2, C_{21}, C_{22}, l_2)) =$$
$$\tfrac{1}{2}(b_1 \sim b_2)(sim(C_{11}, C_{21}) + sim(C_{12}, C_{22}))$$

Let c_p be the set of patterns that fell on a prototype p during the Discovery Phase. We compute the label of p as the sum of the labels of the patterns in c_p, weighted with their respective similarities to p:

$$label(p) \;=\; \sum_{p' = (b, C_1, C_2, l) \,\in\, c_p} \frac{l \cdot sim(p, p')}{|c_p|}$$

To classify a pattern in the Testing Phase, we first determine its prototype. If there is no prototype within the distance θ, the pattern receives the label $-\infty$. Else, we calculate its label as the product of the similarity to the prototype and the label of the prototype.

2.3.2 SVM

To generalize patterns by an SVM, the patterns have to be translated to real-valued feature vectors. For this purpose, we first group the patterns by their bridge ids. Each group will be treated separately so that it is not necessary to store the bridge id in the feature vector. If n is the number of connector symbols, then a feature vector for a pattern can be depicted as follows:

label	context 1			context 2		
R	$X \ldots X$	$\ldots$	$X \ldots X$	$X \ldots X$	$\ldots$	$X \ldots X$
	connector_1		connector_n	connector_1		connector_n

The vector consists of three parts. The first part is the label ($+1$ or -1), which occupies one dimension in the vector as a real value (denoted by R in the scheme above).

The second part and the third part store the context of the first and second placeholder, respectively. Each context contains a sub-part for each possible connector symbol. Each of these subparts contains one bit (denoted by X in the above scheme) for each possible word type. So if there are t word types, the overall length of the vector is $1 + n \cdot t + n \cdot t$. We encode a context as follows in the vector: If there is a link with connector con that points to a word w, we first select the sub-part that corresponds to the connector symbol con. Within this sub-part, we set all bits to 1 that correspond to a type that w has.

The vectors are still grouped according to the bridges. After the Discovery Phase, we pass each group separately to an SVM. We used SVMLight [18] with its default parameters. The SVM produces a *model* for each group, i.e. basically a function from patterns to real values (negative values for negative patterns and positive values for positive ones). To classify a new pattern in the Testing Phase, we first identify its bridge group. If the pattern does not belong to a known group, we give it the label $-\infty$. Else, we translate the pattern to a feature vector and then apply the model of its group. Note that both the kNN classifier and the SVM classifier output a real value that can be interpreted as the confidence of the classification. Thus, it is possible to rank the output pairs by their confidence.

3. EXPERIMENTS

3.1 Setup

We implemented our approach in a system called LEILA (Learning to Extract Information by Linguistic Analysis). We ran LEILA on different corpora with increasing heterogeneity[4]: Wikicomposers (all 872 Wikipedia articles about composers), Wikigeography (all 313 Wikipedia pages about the geography of countries), Wikigeneral (78141 random Wikipedia articles) and Googlecomposers (492 documents as delivered by a Google "I'm feeling lucky" search for composers' names). Since the querying for Googlecomposers was done automatically, this corpus includes spurious advertisements as well as pages with no proper sentences at all.

We tested LEILA on different target relations with increasing complexity: The `birthdate`-relation (e.g. "Chopin" / "1810"), the `synonymy`-relation (e.g. "UN"/"United Nations") and the `instanceOf`-relation (e.g. "Chopin" / "composer").

We compared LEILA to different **competitors**. We only considered competitors that, like LEILA, extract the information from a corpus without using other Internet sources. We wanted to avoid running the competitors on our own corpora or on our own target relations, because we could not be sure to achieve a fair tuning of the competitors. Hence we ran LEILA on the corpora and the target relations that our competitors have been tested on by their authors. We compare the results of LEILA with the results reported by the authors. Our competitors, together with their respective corpora and relations, are the following: *TextToOnto*[5] can extract (i.a.) the `instanceOf` relation by shallow pattern matching and takes arbitrary HTML documents as input. *Text2Onto* [10] (currently under development) is the successor of TextToOnto. *Snowball* [1] uses the slot-extraction paradigm and has been used with the `headquarters` relation. It was trained on a collection of some thousand documents, but for copyright reasons, we only had access to

---

[4] See [30] for more details on the experimental setup and results

[5] http://www.sourceforge.net/projects/texttonto

the test collection (150 text documents). The *CV-system* [11] uses context to assign a concept to an entity. This approach is restricted to the `instanceOf`-relation, but it can classify instances even if the corpus does not contain explicit definitions. In the original paper, the system was run on a collection of 1880 files from the Lonely Planet Internet site[6].

For the **evaluation**, the output pairs of the system have to be compared to a table of ideal pairs. If O denotes the multi-set of the output pairs and I denotes the multi-set of the ideal pairs, then precision p, recall r, and their harmonic mean $F1$ can be computed as

$$r = \frac{|O \cap I|}{|I|} \qquad p = \frac{|O \cap I|}{|O|} \qquad F1 = \frac{2 \times r \times p}{r + p}$$

We estimated precision and recall by extracting the ideal pairs manually for a sub-portion of the corpora. We report confidence intervals for the estimates for $\alpha = 95\%$ (see [30] for details). We measure precision at different levels of recall and report the values for the best F1 value. We use the original *Ideal Metric* for Snowball (see [1]) and the *Relaxed Ideal Metric* for the CV-system (see [30]).

3.2 Results

Table 1 summarizes our experimental results with LEILA on different relations. For the **birthdate** relation, we used Edward Morykwas' list of famous birthdays[7] as examples. LEILA performed very well. For the **synonymy** relation we used all pairs of proper names that share the same synset in WordNet as examples (e.g. "UN"/"United Nations"). As counterexamples, we chose all pairs of nouns that are not synonymous in WordNet (e.g. "rabbit"/"composer"). LEILA performed well.

For the **instanceOf** relation, we used all pairs of a proper name and its lowest non-compound super-concept from WordNet as examples. We used all pairs of a common noun and an incorrect super-concept from WordNet as counterexamples. Our tough evaluation policy lowered LEILA's results: Our ideal pairs include pairs deduced by resolving semantic ambiguities, which decreases LEILA's recall. Furthermore, our evaluation policy demands that non-defining concepts like `friend`, `member` or `successor` not be chosen as instance concepts, which decreases LEILA's precision. Thus, compared to the gold standard of humans, the performance of LEILA can be considered reasonably good.

To test whether heterogeneity influences LEILA, we ran it on the Wikigeneral corpus and finally on the Googlecomposers corpus. The performance dropped in these increasingly challenging tasks, but LEILA could still produce useful results.

Table 2 shows the results for comparing LEILA against various competitors (with LEILA in boldface). Text2Onto seems to have a precision comparable to ours, although the small number of found pairs does not allow a significant conclusion. Both Text2Onto and TextToOnto have drastically lower recall than LEILA.

For Snowball, we only had access to the test corpus. Hence we trained LEILA on a small portion (3%) of the test documents and tested on the remaining ones. LEILA showed a very high precision and a good recall – even though Snowball was trained on a much larger training collection.

For the CV-System, we first used the Lonely Planet corpus as in the original paper [11]. Since the explicit definitions that our system relies on were sparse in the corpus, LEILA performed worse than the competitor. In a second

experiment, we had the CV-system run on the Wikicomposers corpus. This time, our competitor performed worse, because our ideal table is constructed from the definitions in the text, which the CV-system is not designed to follow.

4. CONCLUSION AND OUTLOOK

We proposed to extend the pattern matching approach for information extraction by using deep linguistic structures instead of shallow text patterns. We showed how deep linguistic structures can be represented suitably for machine learning. We proved that the problem of false samples does not question the pattern matching approach. We implemented our approach and we demonstrated that our system LEILA outperforms existing competitors.

Our current implementation leaves room for future work. For example, the linkages allow for more sophisticated ways of resolving anaphoras or matching patterns. LEILA could learn numerous interesting relations (e.g. `country / president` or `isAuthorOf`) and build up an ontology from the results with high confidence. LEILA could acquire and exploit new corpora on its own (e.g. read newspapers) and it could use its knowledge to acquire and structure its new knowledge more efficiently. We plan to exploit these possibilities in our future work.

4.1 Acknowledgements

5. REFERENCES

[1] E. Agichtein, L. Gravano. *Snowball*: extracting relations from large plain-text collections. *ACM 2000*.

[2] J. Aslam, S. Decatur. On the sample complexity of noise-tolerant learning. *Information Processing Letters 1996*.

[3] S. Brin. Extracting patterns and relations from the world wide web. *WWW 1999*.

[4] P. Buitelaar, D. Olejnik, M. Sintek. A protege plug-in for ontology extraction from text based on linguistic analysis. *ESWS 2004*.

[5] P. Buitelaar, S. Ramaka. Unsupervised ontology-based semantic tagging for knowledge markup. *Workshop on Learning in Web Search at ICML 2005*.

[6] R. C. Bunescu, R. Mooney. A Shortest Path Dependency Kernel for Relation Extraction. *EMNLP 2005*.

[7] M. Califf, R. Mooney. Relational learning of pattern-match rules for information extraction. *ACL-97 Workshop in Natural Language Learning 1997*.

[8] V. Cherkassky, M. Yunqian. Practical selection of SVM parameters and noise estimation for SVM regression. *Neural Networks 2004*

[9] P. Cimiano, G. Ladwig, S. Staab. Gimme the context: Contextdriven automatic semantic annotation with cpankow. *WWW 2005*.

[10] P. Cimiano, J. Völker. Text2onto - a framework for ontology learning and data-driven change discovery. *NLDB 2005*.

[11] P. Cimiano, J. Völker. Towards large-scale, open-domain and ontology-based named entity classification. *RANLP 2005*.

[12] O. Etzioni, M. Cafarella, D. Downey, S. Kok, A. Popescu, T. Shaked, S. Soderland, D. S. Weld, A. Yates. Web-scale information extraction in knowitall (preliminary results). *WWW 2004*.

---

[6]http://www.lonelyplanet.com/
[7]http://www.famousbirthdates.com

Table 1: Results with different relations

Corpus	Relation	System	#D	#O	#C	#I	Precision	Recall	F1	%E
Wikicomposers	birthdate	Leila(SVM)	87	95	70	101	73.68% ± 8.86%	69.31% ± 9.00%	71.43%	4.29%
Wikicomposers	birthdate	Leila(kNN)	87	90	70	101	78.89% ± 8.43%	70.30% ± 8.91%	74.35%	4.23%
Wikigeography	synonymy	Leila(SVM)	81	92	74	164	80.43% ± 8.11%	45.12% ± 7.62%	57.81%	5.41%
Wikigeography	synonymy	Leila(kNN)	81	143	105	164	73.43% ± 7.24%	64.02% ± 7.35%	68.40%	4.76%
Wikicomposers	instanceOf	Leila(SVM)	87	685	408	1127	59.56% ± 3.68%	36.20% ± 2.81%	45.03%	6.62%
Wikicomposers	instanceOf	Leila(kNN)	87	790	463	1127	58.61% ± 3.43%	41.08% ± 2.87%	48.30%	7.34%
Wikigeneral	instanceOf	Leila(SVM)	287	921	304	912	33.01% ± 3.04%	33.33% ± 3.06%	33.17%	3.62%
Googlecomposers	instanceOf	Leila(SVM)	100	787	210	1334	26.68% ± 3.09%	15.74% ± 1.95%	19.80%	4.76%
Googlecomposers	instanceOf	Leila(kNN)	100	840	237	1334	28.21% ± 3.04%	17.77% ± 2.05%	21.80%	8.44%
Googlec.+Wikic.	instanceOf	Leila(SVM)	100	563	203	1334	36.06% ± 3.97%	15.22% ± 1.93%	21.40%	5.42%
Googlec.+Wikic.	instanceOf	Leila(kNN)	100	826	246	1334	29.78% ± 3.12%	18.44% ± 2.08%	22.78%	7.72%

#O – number of output pairs   #D – number of documents in the hand-processed sub-corpus
#C – number of correct output pairs   %E – proportion of example pairs among the correct output pairs
#I – number of ideal pairs   Recall and Precision with confidence interval at $\alpha = 95\%$

Table 2: Results with different competitors

Corpus	M	Relation	System	#D	#O	#C	#I	Precision	Recall	F1
Snowball corp.	S	headquarters	Leila(SVM)	54	92	82	165	**89.13%± 6.36%**	**49.70%± 7.63%**	**63.81%**
Snowball corp.	S	headquarters	Leila(kNN)	54	91	82	165	**90.11%± 6.13%**	**49.70%± 7.63%**	**64.06%**
Snowball corp.	S	headquarters	Snowball	54	144	49	165	34.03%± 7.74%	29.70%± 6.97%	31.72%
Snowball corp.	I	headquarters	Leila(SVM)	54	50	48	126	**96.00%± 5.43%**	**38.10%± 8.48%**	**54.55%**
Snowball corp.	I	headquarters	Leila(kNN)	54	49	48	126	**97.96%± 3.96%**	**38.10%± 8.48%**	**54.86%**
Snowball corp.	I	headquarters	Snowball	54	64	31	126	48.44%±12.24%	24.60%± 7.52%	32.63%
Wikicomposers	S	instanceOf	Leila(SVM)	87	685	408	1127	**59.56%± 3.68%**	**36.20%± 2.81%**	**45.03%**
Wikicomposers	S	instanceOf	Leila(kNN)	87	790	463	1127	**58.61%± 3.43%**	**41.08%± 2.87%**	**48.30%**
Wikicomposers	S	instanceOf	Text2Onto	87	36	18	1127	50.00%	1.60%± 0.73%	3.10%
Wikicomposers	S	instanceOf	TextToOnto	87	121	47	1127	38.84%± 8.68%	4.17%± 1.17%	7.53%
Wikicomposers	R	instanceOf	Leila(SVM)	87	336	257	744	**76.49%± 4.53%**	**34.54%± 3.42%**	**47.59%**
Wikicomposers	R	instanceOf	Leila(kNN)	87	367	276	744	**75.20%± 4.42%**	**37.10%± 3.47%**	**49.68%**
Wikicomposers	R	instanceOf	CV-system	87	134	30	744	22.39%	4.03%± 1.41%	6.83%
Lonely Planet	R	instanceOf	Leila(SVM)	–	159	42	289	**26.42%± 6.85%**	**14.53%± 4.06%**	**18.75%**
Lonely Planet	R	instanceOf	Leila(kNN)	–	168	44	289	**26.19%± 6.65%**	**15.22%± 4.14%**	**19.26%**
Lonely Planet	R	instanceOf	CV-system	–	289	92	289	31.83%± 5.37%	31.83%± 5.37%	31.83%

M – Metric (S: Standard, I: Ideal Metric, R: Relaxed Ideal Metric). Other abbreviations as in Table 1

[13] A. Finn, N. Kushmerick. Multi-level boundary classification for information extraction. *ECML 2004*.

[14] D. Freitag, N. Kushmerick. Boosted wrapper induction. *American Nat. Conf. on AI 2000*.

[15] J. Graupmann. Concept-based search on semi-structured data exploiting mined semantic relations. *EDBT Workshops 2004*.

[16] A. Hearst. Automatic acquisition of hyponyms from large text corpora. *ICCL 1992*.

[17] F. C. J. Iria. Relation extraction for mining the semantic web, 2005.

[18] T. Joachims. *Learning to Classify Text Using Support Vector Machines*. PhD thesis, Dortmund, 2002.

[19] M. J. Kearns, R. E. Schapire. Efficient distribution-free learning of probabilistic concepts. *Computational Learning Theory and Natural Learning Systems*, 1994.

[20] D. Lin, P. Pantel. Dirt: Discovery of inference rules from text. *KDD 2001*.

[21] A. Maedche, S. Staab. Discovering conceptual relations from text. *ECAI 2000*.

[22] R. Montague. Universal grammar. *Formal Philosophy. Selected Papers of Richard Montague.* Yale University Press, 1974.

[23] D. Ravichandran, E. Hovy. Learning surface text patterns for a question answering system. *ACL 2002*.

[24] E. Riloff. Automatically generating extraction patterns from untagged text. *Annual Conf. on AI 1996*.

[25] H. U. Simon. General bounds on the number of examples needed for learning probabilistic concepts. *COLT 1993*.

[26] D. Sleator, D. Temperley. Parsing english with a link grammar. *Int. Workshop on Parsing Technologies 1993*.

[27] R. Snow, D. Jurafsky, A. Y. Ng. Learning syntactic patterns for automatic hypernym discovery. *NIPS 2005*.

[28] S. Soderland. Learning information extraction rules for semi-structured and free text. *Machine Learning*, pages 233–272, 1999.

[29] F. M. Suchanek, G. Ifrim, G. Weikum. Combining Linguistic and Statistical Analysis to Extract Relations from Web Documents. *Technical Report* MPI-I-2006-5-004 at the Max-Planck-Institute for Computer Science Saarbrücken/Germany.

[30] F. M. Suchanek, G. Ifrim, G. Weikum. LEILA: Learning to Extract Information by Linguistic Analysis. *OLP Workshop at COLING/ACL 2006*

[31] S. Soderland, D. Fisher, J. Aseltine, W. Lehnert. Crystal: Inducing a conceptual dictionary. *IJCAI 1995*.

[32] F. Xu, H. U. Krieger. Integrating shallow and deep NLP for information extraction. *RANLP 2003*.

[33] F. Xu, D. Kurz, J. Piskorski, S. Schmeier. Term extraction and mining term relations from free-text documents in the financial domain. *BIS 2002*.

[34] R. Yangarber, R. Grishman, P. Tapanainen, S. Huttunen. Automatic acquisition of domain knowledge for information extraction. *ICCL 2000*.

[35] R. Yangarber, W. Lin, R. Grishman. Unsupervised learning of generalized names. *ICCL 2002*.

[36] L. Zhang, Y. Yu. Learning to generate CGs from domain specific sentences. *LNCS*, 2120:44–57, 2001.