Open access • Proceedings Article • DOI:10.1145/1150402.1150492

Combining linguistic and statistical analysis to extract relations from web documents — Source link

Fabian M. Suchanek, Georgiana Ifrim, Gerhard Weikum

Institutions: Max Planck Society

Published on: 20 Aug 2006 - Knowledge Discovery and Data Mining

Topics: Relationship extraction, Natural language and Pattern matching

Combining Linguistic and Statistical Analysis to Extract Relations from Web Documents

Fabian M. Suchanek, Georgiana Ifrim,
Gerhard Weikum

MPI–I–2006–5–004 April 2006

Authors' Addresses

Fabian M. Suchanek
Max-Planck-Institut für Informatik
Stuhlsatzenhausweg 85
66123 Saarbrücken
Germany

Georgiana Ifrim
Max-Planck-Institut für Informatik
Stuhlsatzenhausweg 85
66123 Saarbrücken
Germany

Gerhard Weikum
Max-Planck-Institut für Informatik
Stuhlsatzenhausweg 85
66123 Saarbrücken
Germany

Acknowledgements

We would like to thank Eugene Agichtein for his caring support with Snowball. Furthermore, Johanna Völker and Philipp Cimiano deserve our sincere thanks for their unreserved assistance with their system.

Abstract

Search engines, question answering systems and classification systems alike can greatly profit from formalized world knowledge. Unfortunately, manually compiled collections of world knowledge (such as WordNet or the Suggested Upper Merged Ontology SUMO) often suffer from low coverage, high assembling costs and fast aging. In contrast, the World Wide Web provides an endless source of knowledge, assembled by millions of people, updated constantly and available for free. In this paper, we propose a novel method for learning arbitrary binary relations from natural language Web documents, without human interaction. Our system, LEILA, combines linguistic analysis and machine learning techniques to find robust patterns in the text and to generalize them. For initialization, we only require a set of examples of the target relation and a set of counterexamples (e.g. from WordNet). The architecture consists of 3 stages: Finding patterns in the corpus based on the given examples, assessing the patterns based on probabilistic confidence, and applying the generalized patterns to propose pairs for the target relation. We prove the benefits and practical viability of our approach by extensive experiments, showing that LEILA achieves consistent improvements over existing comparable techniques (e.g. Snowball, TextToOnto).

Keywords

Ontology Learning, Relation Extraction, Information Extraction, Linguistic Analysis

Contents

1 Introduction

1.1 Motivation

Many data mining tasks such as classification, ranking, recommendation, or data cleaning could be boosted by explicit background knowledge in the form of ontologies (e.g., SUMO [29]), thesauri (e.g., WordNet [15]), or lexicons. Unfortunately, the manual construction and maintenance of such knowledge bases is a limiting factor in our modern world of "exploding information". Recently, various projects have pursued ways of utilizing the World Wide Web and other poorly structured information sources for automatically creating ontological relations in an almost unsupervised manner. These projects include early, small-scale approaches like Dipre [4], Snowball [1], and TextToOnto [11] as well as very recent projects like KnowItAll [14] and KnowItNow [7] that aim at large-scale knowledge discovery and harvesting on the Web. In this context, knowledge acquisition amounts to finding as many instances as possible for unary or binary semantic relations such as `Cities(x)`, `Scientists(x)`, `Headquarters(company, city)`, `BirthDates(person, date)`, or `Plays(person, instrument)`, including generic relations like `InstanceOf(entity, class)`.

At the heart of such knowledge acquisition projects are NLP (Natural Language Processing) and text mining techniques. Prior approaches have limited the NLP part to part-of-speech tagging [26] and focused mostly on matching textual surface patterns such as "`x such as y`" (one of the Hearst patterns [19]), in combination with machine learning techniques and statistical inferences for assessing the validity of newly discovered patterns and relation instances. The actual NLP and text analysis parts have been restricted in their expressiveness to regular expression matching on text sequences. To our knowledge, none of the prior work considered utilizing deeper linguistic analysis such as constructing NLP parse trees or even graph structures and running matching and learning methods on these richer representations. Deeper linguistic analysis seems to be the key for improving both precision and recall of unsupervised knowledge acquisition from

corpora like the Web or textual lexicons such as Wikipedia or Encarta.

1.2 Related Work

There are numerous Information Extraction (IE) approaches, which differ in various features:

- **Type of the extracted relation:** The extracted relations can be either unary or binary. In the unary case, the relations are just lists of entities (e.g. all `cities` in a given text, [16, 8]). In this paper we focus on binary relations (e.g. the `birthdate`-relation, which holds between a person and her birthdate). Some systems are designed to discover new binary relations ([25]). However, we assume that the user gives the system the target relation he is interested in. Some systems are restricted to learning a single relation, mostly the `instanceOf`-relation ([12, 5]). In this paper, we are interested in extracting arbitrary relations. This not only includes the `instanceOf`-relation, but also relations like the `birthdate`-relation or the `headquarters`-relation between a company and the city of its headquarters.

- **Human interaction:** There are systems that require human input for the IE process ([31]). Our work aims at a completely automated system.

- **Type of corpora:** There exist systems that can extract information efficiently from formatted data, such as HTML-tables or structured text ([18, 17]). However, since a large part of the Web consists of natural language text, we consider in this paper only systems that accept also unstructured corpora.

- **Initialization:** As initial input, some systems require a hand-tagged corpus ([20, 36]), i.e. a corpus in which the relevant items have been marked manually. Other systems require text patterns ([39]) or templates ([37]), i.e. phrases that indicate a pair of the target relation. Again other systems require seed tuples ([1]), i.e. a list of pairs of the target relation. There is also a class of systems that require just tables of target concepts ([11]). Since hand-labeled data and manually assembled text patterns require huge human effort, we consider only systems that use seed pairs or tables of concepts.

Furthermore, we differentiate between *closed* systems that are bound to a corpus and *open* systems that use the Web as a corpus. *KnowItAll* [14] is an example of an open system. It is instantiated with a set of extraction rules that are used to

generate keyword queries to search engines. Another system that makes use of the Web is [10]. We observe that in both open and closed systems, the techniques used to extract the entities from the documents are essential. We concentrate in this paper on this type of techniques; to study them in a controlled environment, we restrict ourselves to closed systems for this paper.

There are many different **techniques** for extracting entities from documents. One school concentrates on detecting the boundary of interesting entities in the text, [8, 16, 40]). This usually goes along with the restriction to unary target relations. Other approaches make use of the context in which an entity appears ([11, 6]). This school is mostly restricted to the `instanceOf`-relation. The only group that can learn arbitrary binary relations is the group of pattern matching systems ([14, 1, 30, 4, 35, 38]). Surprisingly, none of these systems uses deep linguistic analysis of the corpus. Consequently, most of them are extremely volatile to small variations in the patterns – even if the variation does not have any semantic effect. For example, the simple subordinate clause in the following example (taken from [30]) can already prevent a surface pattern matcher from discovering the relation between "London" and the "river Thames": "London, which has one of the busiest airports in the world, lies on the banks of the river Thames."

1.3 Contribution

This paper presents LEILA (Learning to Extract Information by Linguistic Analysis), a system with novel techniques for richer acquisition of binary relations from Web and text documents. LEILA uses a link-grammar representation [34] for natural-language sentences as well as other advanced NLP methods like anaphora resolution, and combines them with statistical learning for robust and high-yield information extraction. Our experimental studies on a variety of corpora demonstrate that LEILA achieves very good results in terms of precision and recall and clearly outperforms the prior state-of-the-art methods. The paper's novel contributions are:

- We show how advanced NLP techniques like link grammars and anaphora resolution can be harnessed for richer representation of natural-language sentences and more expressive detection of semantic relations.
- We develop a feature model for the link-grammar-based graph representation of a sentence that is expressive enough to capture patterns beyond the previous state of the art but, at the same time is robust to avoid overfitting and efficiently tractable.

- Based on this feature model we design statistical learners, using SVM or kNN classifiers, that can discriminate good versus bad patterns for a given target relation.
- All our techniques are carefully integrated into a full-fledged system architecture and implemented in our LEILA system.

2 System Model

2.1 Link Grammars

There are different approaches for parsing natural language sentences. It is possible to use just regular expressions to discover chunks of related words or to assign part-of-speech tags, but we already argued for a more detailed analysis. Often, context-free grammars are used and a number of parsers are available to construct context-free parse-trees for natural language sentences ([23]). More advanced techniques use non-context-free feature structures instead of simple parse trees. These include Lexical Functional Grammar parsers ([27]) or Head-Driven Phrase Structure Grammar parsers ([2]). These techniques have been extended by stochastic models, resulting in ever more robust, but also more complex parsers ([26, 13]).

For our implementation, we chose the Link Grammar Parser [34]. 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[1]. Figure 2.1 shows a simplified example of a linguistic structure produced by the link parser:

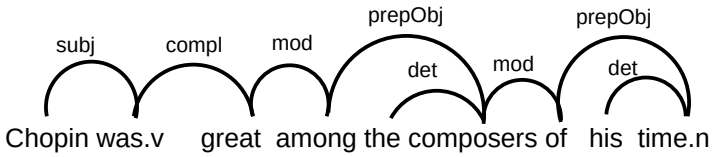

Figure 2.1: A simple linkage

We call these structures *linkages*. Formally speaking, a linkage is a connected planar undirected graph, the nodes of which are the words of the sentence. The edges are called *links*. They are labeled with *connectors*, taken from a finite set

[1] [24] and [41] use the same parsing technique, albeit for different purposes.

of symbols. 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*. A link grammar is a set of rules that specify which word may be linked by which connector to preceding and following words. For example, the link grammar may specify that the word "was" has to have a `subj`-link to a preceding word and a `compl`-link to a following word. The parser also assigns *part-of-speech tags* to the words, i.e. symbols identifying the grammatical function of the words. In the example shown in Figure 2.1, the letter "n" following the word "composers" identifies "composers" as a noun.

Figure 2.2 shows how the Link Parser copes with a more complex example. The relationship between the subject "London" and the verb "lies" is not disrupted by the subordinate clause: For ambiguous sentences, the Link Parser gen-

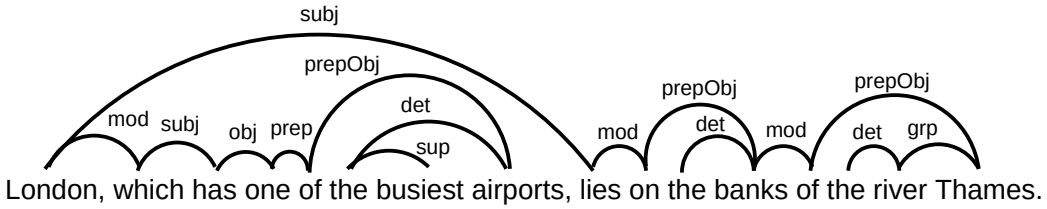

Figure 2.2: A complex linkage

erates multiple linkages. When faced with an erroneous sentence, the Link Parser tries to ignore some grammatical constraints in order to find a linkage anyway. Such a linkage is assigned a heuristic cost based on the number of constraints that have been violated. In the end, the parser outputs the linkages in ascending order of their cost.

We say that a linkage *expresses* a relation r, if the underlying sentence implies that a pair of entities is in r. For example, the linkage in Figure 2.2 expresses the `possession`-relation, because it states that "London" has an "airport". 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 ([28]). For this paper, however, we limit ourselves to an intuitive understanding of the notion of meaning.

We define a *pattern* as a linkage in which two words have been replaced by placeholders. Figure 2.3 shows a pattern derived from the linkage in Figure 2.1 by replacing "Chopin" and "composers" by the placeholders "X" and "Y". We call the (unique) shortest path from one placeholder to the other the *bridge*, marked in bold in the figure. 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 above pattern matches the linkage in Figure 2.4, because the bridge of the pattern occurs in the linkage, apart from a substitution of "great" by "mediocre".

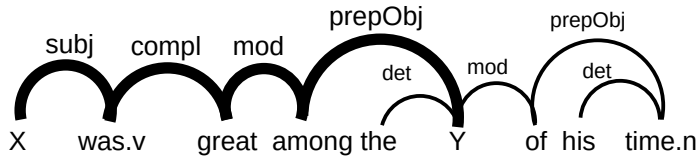

Figure 2.3: A pattern

If a pattern matches a linkage, we say that the pattern *produces* the pair of words

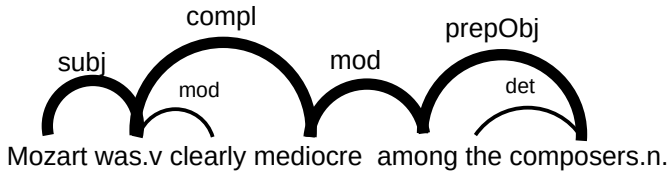

Figure 2.4: A matching linkage

that the linkage contains in the position of the placeholders. In the example in Figure 2.4, the pair ”Mozart” / ”composers” is produced.

2.2 Algorithm

As a definition of the target relation, our algorithm requires a function (given by a Java method) that decides into which of the following categories a word pair falls:

- The pair can be an ***example*** for the target relation. E.g., for the `birthdate`-relation, a list of persons with their birth dates can serve as examples.
- The pair can be a ***counterexample*** for the target relation. E.g., if the pair ”Chopin” / ”1810” is in the example list, then the pair ”Chopin” / ”2000” must be a counterexample.
- The pair can be a ***candidate*** for the target relation. For the `birthdate`-relation, only pairs of a proper name and a date are candidates.
- None of the above.

The corpus should be a sequence of natural language sentences. These sentences are parsed, producing a deep *grammatical structure* [26] 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. For our implementation, however, we used the Link Grammar structures introduced above.

Our algorithm proceeds in three phases:

1. In the *Discovery Phase*, it seeks sentences in which an example appears. In the corresponding linkage, the two words are replaced by placeholders, resulting in a pattern. The patterns collected this way are called *positive patterns*.

2. In the second phase, the *Assessment Phase*, the algorithm finds all linkages that match a positive pattern, but produce counterexamples. The corresponding patterns are collected as *negative patterns*. Now, statistical learning is applied to learn the concept of positive patterns from the positive and the negative patterns. The result of this process is a classifier, i.e. a function from patterns to boolean values.

3. In the last phase, the *Harvesting 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 are a candidate and the pattern is classified as positive, the produced pair is proposed as a new element of the target relation. These new pairs are called the *output pairs* of the algorithm.

For testing purposes, it is possible to run the Harvesting Phase on a different corpus. In this case, we refer to the Discovering Phase and the Assessment Phase collectively as *Training*, whereas the Harvesting Phase is also referred to as *Testing*.

2.3 Statistical Model

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 call these patterns *false samples*.

Our approach is not bound to a certain machine learning algorithm, but virtually any learning algorithm that can deal with a limited number of false samples is suitable. For Support Vector Machines (SVM), the effect of false samples has been analyzed thoroughly [9]. In general, SVM is highly tolerant to noise. There are also detailed theoretical studies [3] 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 [22]. In this setting, the pattern is not 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. [33] 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.

2.3.1 k-Nearest-Neighbor Classifiers

A k-Nearest-Neighbors (kNN) classifier requires a distance function on the patterns. During training, positive and negative patterns are collected. In the testing phase, a pattern is classified as positive iff the (distance-weighted) majority of its k nearest neighbors is positive. We consider a simple variant of an adaptive kNN classifier. Based on the distance function, we cluster the sample patterns into classes. We show in Section 3.2 how to establish a distance function on patterns for this clustering. With our distance function, the clusters are equivalence classes, i.e. all patterns in the class match the same linkages and we may assume that they all have the same probability of producing an example or a counterexample. When a new pattern needs to be classified, we first determine its equivalence class. If the majority of the patterns in the class is positive, we classify the new pattern as positive, else as negative.

We now consider the problem of false samples. We concentrate on false positives, as the problem of false negatives is dual. We analyze the probability that one given equivalence class E classifies a new pattern as positive, although the patterns in E do not express the target relation. Since all patterns in E share the same properties, we occasionally refer to E as one single pattern.

We first concentrate on the probability of E containing more positive patterns than negative patterns, although E does not express the target relation. We model the sentences as a sequence of N random events. For each sentence, we can have three types of events: (1) E matches the linkage and produces an example, (2) E matches and produces a counterexample or (3) neither. We describe these three events by Bernoulli random variables A, B, C, captured by a multinomial distribution: $A = 1$ with probability p_A iff an example is produced, $B = 1$ with probability p_B iff a counterexample is produced and $C = 1 - A - B$ with proba-

bility $p_C = 1 - p_A - p_B$. The key assumption is that $p_A < p_B$, since E does not express the target relation. Let $\#A$ stand for the number of produced examples and $\#B$ for the number of counterexamples. We are interested in the probability of E being a false positive, namely $P(\#A > \#B)$, given that $p_A < p_B$ ($p_C > 0$). In the appendix, we prove the upper bound (using Chernoff-Hoeffding bounds)

$$P(\#A > \#B) \leq 2e^{-N\frac{(1-p_C)^2}{2}} + 2e^{-(N(1-p_C)+2)(\frac{1}{2}-\tilde{p_A})^2}$$

where $\tilde{p_A} = \frac{p_A}{p_A+p_B}$. Now, we concentrate on the probability that a new pattern falls into E, $P(E)$. We estimate this probability as a multiple of $1 - p_C$: $P(E) = \beta(1 - p_C)$ for some $\beta \geq 1$. The better the examples and counterexamples are chosen, the smaller β will be (in our experiments, $\beta \approx 1.69$). Then the probability that E classifies a new pattern wrongly is bounded by

$$2\beta(1 - p_C)(e^{-N\frac{(1-p_C)^2}{2}} + e^{-(N(1-p_C)+2)(\frac{1}{2}-\tilde{p_A})^2})$$

This estimation mirrors the intuition that either a negative equivalence class is rare (p_C is large) and then it rarely matches in the Harvesting Phase or the class is frequent (p_C is small) and then the probability of containing too many positive patterns is small.

3 Implementation

3.1 Document Preprocessing

In order to allow our system LEILA to learn relations involving dates and numbers, we normalize date and number expressions by regular expression matching. For example, the expression "November 23rd to 24th 1998" becomes "1998-11-23 to 1998-11-24" and the expression "0.8107 acre-feet" becomes "1000 cubic-meters".

LEILA accepts, but is not restricted to, HTML documents. Our preprocessing produces two files from the original document: The first file contains the proper sentences with the HTML-tags removed. The second file contains the non-grammatical parts, such as lists and expressions using parentheses. For example, the character sequence "Chopin (born 1810) was a great composer" is split into the sentence "Chopin was a great composer" and the non-grammatical information "Chopin (born 1810)". A list like "Some well-known composers are: <UL><LI> Chopin <LI> Mozart... </UL>" produces the following output for the non-grammatical file:

"Some well-known composers are: # Chopin"
"Some well-known composers are: # Mozart"
...

We give the files with the proper sentences to the Link Grammar parser. As it comes with no additional computational cost, we have the parser produce its three most likely linkages for each sentence. For the non-grammatical files, we provide a pseudo-parsing, which simply links each two adjacent words by an artificial connector. As a result, the uniform output of the preprocessing is a sequence of linkages.

We use a very basic form of named entity recognition. First, we concatenate all words that are joined by "<A> ... </A>" tags. Next, we use the fact that the Link Parser links noun groups like "Frederic Chopin" or "United States

of America" by designated connectors. We join words that are linked by these connectors. For our goal, it is essential to normalize nouns to their singular form. This task is non-trivial, because there are numerous words with irregular plural forms and there exist even word forms that can be either the singular form of one word or the plural form of another. By collecting these exceptions systematically from WordNet, we were able to stem most of them correctly with our Plural-to-Singular Stemmer (*PlingStemmer*[1]).

Anaphora resolution increases the number of sentences that LEILA can use. Although the linkages would allow for quite sophisticated techniques, we restrict ourselves to a conservative approach for the time being: We replace a third person pronoun by the subject of the preceding sentence, if the singular-vs-plural forms match. Furthermore, the system uses a simple form of regular expression matching to detect possible person names and company names. This allows to resolve company references (like "the company") to the corresponding company name.

3.2 Feature Model

This section discusses how patterns can be represented and generalized. It would obviously not be too difficult to somehow encode the full linkages of patterns into feature vectors. However, such an approach would not generalize well, for it would capture all details of the specific sentences that led to the patterns and thus tend to cause overfitting. So the problem that we tackle is to identify the characteristic but generalized features within linkages as training input for the statistical learner. 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*. Furthermore, each pattern is given a *label*: 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 the placeholder "Y" in the pattern of Figure 2.3 is the set of triples {(`det`, `left`, "the"), (`prepObj`, `left`, "among"), (`mod`, `right`, "of")}. We distinguish the following *types* of words: Nouns, adjectives, prepositions, verbs, numbers, dates, names, person names, company names and abbreviations. A word can have a set of corresponding types. 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`. Furthermore, we maintain a list of stopwords. To accommodate the considerable role that stopwords play in the understanding of a sentence, we make each stop-

[1] http://www.mpii.mpg.de/$\sim$ suchanek/personal/programs/javaexport

word 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 2.3 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 for LEILA: One is the simple adaptive kNN classifier discussed in 2.3.1 and the other one uses SVM.

3.2.1 kNN

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

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

Let $\tau(w)$ be the set of types of a word w. We define the following similarity functions for words w_1, w_2, contexts C_1, C_2 and patterns $(b_1, C_{11}, C_{12}, l_1), (b_2, C_{21}, C_{22}, l_2)$:

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

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

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

where $\alpha_1,\ \alpha_2,\ \alpha_3$ are weighting factors that sum up to 1. We chose $\alpha_1 = 0.4,\ \alpha_2 = 0.2,\ \alpha_3 = 0.4$. We consider all patterns p_1, p_2 with $sim(p_1, p_2) > \theta$ to be in the same equivalence class for some real value θ. If θ is large, memory consumption increases and precision becomes better, but the generalization suffers. We chose $\theta = 0.5$. We store an equivalence class simply by storing a

prototype pattern. If we see a new pattern that does not fall into an existing equivalence class, it becomes the prototype for a new equivalence class. At the end of the Assessment Phase, the label of an equivalence class c with its prototype c_p is computed as the average value of the labels of the patterns in the class, weighted with their respective similarities to the prototype:

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

To classify a previously unseen pattern, we first determine its equivalence class. Then we calculate its label as the product of its similarity to the prototype and the label of the equivalence class.

3.2.2 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 can be depicted as follows:

$$\overbrace{R}^{\text{label}} \overbrace{\underbrace{X \ldots X}_{\text{connector}_1} \quad \ldots \quad \underbrace{X \ldots X}_{\text{connector}_n}}^{\text{context 1}} \overbrace{\underbrace{X \ldots X}_{\text{connector}_1} \quad \ldots \quad \underbrace{X \ldots X}_{\text{connector}_n}}^{\text{context 2}}$$

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 $1 + n \times t + n \times t$. We encode a context as follows in the vector: If there is a link of 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 and the Assessment Phase, we pass each group separately to an SVM. We used Thorsten Joachims' SVMLight [21] with its default parameters. The SVM produces a *model* for each group, which allows it to classify previously unseen vectors. To classify a new pattern as positive or negative, we first identify the group it belongs to. We translate the pattern to a vector. Then we use the model

corresponding to the group to classify the vector. Both the kNN and SVM classifiers output a real value that can be interpreted as the confidence of classifying a test sample and can be used for computing precision at different levels of recall.

4 Experiments

4.1 Setup

We ran LEILA on different **corpora**:

- **Wikicomposers**. This is the set of all Wikipedia articles[1] about composers. We use it to see how LEILA performs on a document collection with a strong structural and thematic homogeneity. We downloaded all documents that are listed in Wikipedia's list of composers. The set consists of 872 HTML documents.

- **Wikigeneral**. The set of all Wikipedia articles starting with "G" or "M". We chose it to assess LEILA's performance on structurally homogenous, but thematically random documents. The set contains 78141 HTML documents.

- **Wikigeography**. The set of all Wikipedia pages about the geography of countries. This set contains 313 HTML documents.

- **Googlecomposers**. This is a set of web pages about composers obtained by querying the search engine Google. We use it to see how LEILA performs on a corpus with a high structural heterogeneity. We queried Google for each composer name that appeared in Wikipedia's composer list. We restricted ourselves to the baroque, classical, and romantic composers. For each composer name, we downloaded the HTML page that ranked highest in the query result. If the highest ranked page was a Wikipedia article, we chose the second page. The set contains 492 HTML documents. Since the querying was done automatically, a downloaded page is not necessarily

[1] http://www.wikipedia.org

about the composer we queried for. The pages include spurious advertisements as well as pages with no proper sentences at all.

We tested LEILA with different **target relations**. These include

- The `birthdate` relation, which holds between a person and his birthdate (for example "Chopin"/"1810"). This relation is easy to learn, because it is bound to strong surface clues (the first element is always a name, the second is a date). Furthermore, the likelihood of false positives is small, because it is extremely unlikely that a person and her birthdate appear by chance in a sentence.

- The `synonymy` relation, which holds between two names that refer to the same entity (for example "UN"/"United Nations"). This relation is more sophisticated, since there are no surface clues.

- The `instanceOf` relation, which holds between an entity and its concept (for example "Chopin"/"composer"). This relation is even more sophisticated, because the sentences often express it only implicitly.

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 ensure 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. The following list enumerates our competitors, together with their respective corpora and relations:

- **TextToOnto**. This is a representative of methods that use surface patterns. This approach has been perfected especially for the `instanceOf` relation. Hence we chose as competitor a state-of-the-art pattern matcher for this relation, TextToOnto[2]. For completeness, we also consider its successor Text2Onto [11], although it contains only default methods in its current state of development.

- **Snowball**. To compare our system with the classical slot-extraction paradigm, we chose Snowball [1] as a recent competitor. In the original paper, Snowball has been tested on the `company/headquarters` relation. It holds between a company and the city of its headquarters. Snowball was trained on a collection of some thousand documents and then applied to a test collection. For copyright reasons, we only had access to the test collection. It consists of 150 text documents.

[2]http://www.sourceforge.net/projects/texttoonto

- There is a class of competitors that use context to assign a concept to an entity. These systems are restricted to the `instanceOf`-relation, but they can classify instances even if the corpus does not contain explicit definitions. We examined one of the newest systems in this field: The system designed by Cimiano and Völker [12], which we will refer to as the **CV-system**. In the original paper, the system was tested on a collection of 1880 files from the Lonely Planet Internet site[3].

For the **evaluation**, the output pairs of the system have to be compared to a table of ideal pairs. There are two different ways of defining the ideal pairs: *Ground-truth based* or *document-based*.

For the **ground-truth evaluation technique**, the ideal pairs are a set of ground truth pairs that are independent of the corpus. This method seems inadequate for our purpose for two reasons: First, the ground truth table may be neither a subset or a superset of the facts expressed in the documents. Second, the technique does not allow us to measure the yield of the system with respect to the document content. We are interested in how good the system performs at extracting pairs as compared to a human reader.

The latter question is answered by the **document-based evaluation technique**. For this technique, the ideal pairs are extracted manually from the documents. In our methodology, the ideal pairs comprise all pairs that a human would understand to be elements of the target relation. This involves full anaphora resolution, the solving of reference ambiguities, and the choice of truly defining concepts. For example, we accept Chopin as instance of `composer` but not as instance of `member`, even if the text says that he was a member of some club. Of course, we expect neither the competitors nor LEILA to achieve the results in the ideal table.

There is a variation of the document-based evaluation technique called the *Ideal Metric* [1]. We use the Ideal Metric only to compare LEILA to Snowball, which has been optimized for this metric. The Ideal Metric assumes the target relation to be right-unique (i.e. a many-to-one relation). Hence the ideal pairs are right-unique. The output pairs can be made right-unique by selecting the pair with the highest confidence for each first component. Next, duplicates have to be removed from the ideal pairs and also from the output pairs. Also, one removes all output pairs that have a first component that is not in the ideal set. Intuitively, the Ideal Metric understands the results of the system as a function from the first pair component to the second. It compares the function calculated by the system to the ideal function. Once the ideal pairs are defined, precision, recall, and their harmonic mean $F1$ can be computed as follows, where $Output$ denotes the multi-

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

set of the output pairs and $Ideal$ denotes the multi-set of the ideal pairs:

$$recall = \frac{|Output \cap Ideal|}{|Ideal|} \qquad precision = \frac{|Output \cap Ideal|}{|Output|}$$

$$F1 = \frac{2 \times recall \times precision}{recall + precision}$$

There is one special case for the CV-system, which uses the Ideal Metric for the non-right-unique `instanceOf` relation. To allow for a fair comparison, we used the *Relaxed Ideal Metric*, which does not make the ideal pairs right-unique. The calculation of recall is relaxed as follows:

$$recall = \frac{|Output \cap Ideal|}{|\{x|\exists y : (x, y) \in Ideal\}|}$$

All document-based evaluation techniques require the manual extraction of ideal pairs. Due to the effort, we processed only a small proportion of the documents by hand. To ensure significance in spite of this, we compute confidence intervals for our estimates: We interpret the sequence of output pairs as a repetition of a Bernoulli-experiment, where the output pair can be either correct (i.e. contained in the ideal pairs) or not. The parameter of this Bernoulli-distribution is the precision. We estimate the precision by drawing a sample (on the hand-labeled documents). By assuming that the output pairs are identically independently distributed, we can calculate a confidence interval for our estimation. We report confidence intervals for precision and recall for a confidence level of $\alpha = 95\%$.

For the evaluation, we used approximate string matching techniques to account for different writings of the same entity. For example, we count the output pair "Chopin" / "composer" as correct, even if the ideal pairs contain "Frederic_Chopin" / "composer". We measured precision at different levels of recall and report the values for the best F1 value. To find out whether LEILA just reproduces the given examples, we also report the number of examples among the output pairs. During our evaluation, we found that the Link Grammar parser does not finish parsing on roughly 1% of the files (for major grammatical erros or indigestable input).

4.2 Results

4.2.1 Results on different relations

We tested LEILA on the following target relations: `birthdate`, `synonymy` and `instanceOf`. Table 4.1 summarizes our experimental results.

For the **birthdate** relation, we used Edward Morykwas' list of famous birthdays[4] as examples, and we chose all pairs of person and incorrect birthdate as counterexamples. All pairs with of a proper name and a date are candidates.
We ran LEILA on the Wikicomposer corpus. LEILA performed quite well on this task. The patterns found were of the form "*X* was born in *Y*" and "*X* (*Y*)". The quality of the results decreases as the system starts to consider any number in brackets a birthdate. For example, at the lower end of the confidence scale, the system also reports operas with the date of their first performance.

As examples on **synonymy** we used all pairs of proper names that share the same synset in WordNet. As counterexamples, we chose all pairs of nouns that are not synonymous in WordNet. For instance, "UN"/"United Nations" is an example and "rabbit"/"composer" is a counterexample. All pairs of proper names are candidates.
We ran LEILA on the Wikigeography corpus, because this set is particularly rich in synonyms. LEILA performed reasonably well. The patterns found include "*X* was known as *Y*" as well as several non-grammatical constructions such as "*X* (formerly *Y*)".

The most interesting relation is the `instanceOf` relation. If an entity belongs to a concept, it also belongs to all super-concepts. However, admitting each pair of an entity and one of its super-concepts as an example would have resulted in far too many false positives. In contrast, restricting oneself to the lowest concept might make the system miss useful patterns. The problem is to determine for each entity the (super-)concept that is most likely to be used in a natural language definition of that entity. Psychological evidence [32] suggests that humans prefer a certain layer of concepts in the taxonomy to classify entities. The set of these concepts is called the *Basic Level*. Heuristically, we found that the lowest super-concept in WordNet that is not a compound word is a good approximation of the basic level concept for a given entity. We used all pairs of a proper name and the corresponding basic level concept of WordNet as examples. We could not use pairs of proper names and incorrect super-concepts as counterexamples, because our corpus Wikipedia knows more meanings of proper names than WordNet. Therefore, we used all pairs of common nouns and incorrect super-concepts from WordNet as counterexamples. All pairs of a proper name and a WordNet concept are candidates.

We ran LEILA on the Wikicomposers corpus. The performance on this task was acceptable, but not impressive. However, the chances to obtain a high recall and a high precision were significantly decreased by our tough evaluation policy: The ideal pairs include tuples deduced by resolving syntactic and semantic ambiguities and anaphoras. Furthermore, our evaluation policy demands that non-

[4] http://www.famousbirthdates.com

defining concepts like `member` not be chosen as instance concepts. In fact, a high proportion of the incorrect assignments were `friend`, `member`, `successor` and `predecessor`, decreasing the precision of LEILA. Thus, compared to the gold standard of humans, the performance of LEILA can be considered reasonably good.

The patterns found include the Hearst patterns [19] "*Y* such as *X*", but also more interesting patterns like "*X* was known as a *Y*", "*X* [...] as *Y*", "*X* can be regarded as *Y*" and "*X* is unusual among *Y*".

To test whether thematic heterogeneity influences LEILA, we ran it on the Wikigeneral corpus. Finally, to try the limits of our system, we ran it on the Googlecomposers corpus. As shown in Table 4.1, the performance of LEILAdropped in these increasingly challenging tasks, but LEILAcould still produce useful results.

The different learning methods (kNN and SVM) performed similarly for all relations. Of course, in each of the cases, it is possible to achieve a higher precision at the price of a lower recall. Parsing the files with the Link Parser constitutes the largest part of the run-time (approx. 42 seconds per file, e.g 3:45h for the Wikigeography corpus). Training and testing on all corpora except Wikigeneral is in the range of 2-15 minutes, with the SVM being a bit faster than kNN. The Wikigeneral corpus with its roughly $80,000$ documents takes about 5h for training and testing with the SVM. We did not employ kNN for this corpus.

Table 4.1: Results with different relations

Corpus	Relation	System	#O	#C	#I	Precision	Recall	F1	#E	%E
Wikicomposers	birthdate	LEILA(SVM)	95	70	101	73.68% $\pm$ 8.86%	69.31% $\pm$ 9.00%	71.43%	3	4.29%
Wikicomposers	birthdate	LEILA(kNN)	90	70	101	78.89% $\pm$ 8.43%	70.30% $\pm$ 8.91%	74.35%	3	4.23%
Wikigeography	synonymy	LEILA(SVM)	92	74	164	80.43% $\pm$ 8.11%	45.12% $\pm$ 7.62%	57.81%	4	5.41%
Wikigeography	synonymy	LEILA(kNN)	143	105	164	73.43% $\pm$ 7.24%	64.02% $\pm$ 7.35%	68.40%	5	4.76%
Wikicomposers	instanceOf	LEILA(SVM)	685	408	1127	59.56% $\pm$ 3.68%	36.20% $\pm$ 2.81%	45.03%	27	6.62%
Wikicomposers	instanceOf	LEILA(kNN)	790	463	1127	58.61% $\pm$ 3.43%	41.08% $\pm$ 2.87%	48.30%	34	7.34%
Wikigeneral	instanceOf	LEILA(SVM)	921	304	912	33.01% $\pm$ 3.04%	33.33% $\pm$ 3.06%	33.17%	11	3.62%
Googlecomposers	instanceOf	LEILA(SVM)	787	210	1334	26.68% $\pm$ 3.09%	15.74% $\pm$ 1.95%	19.80%	10	4.76%
Googlecomposers	instanceOf	LEILA(kNN)	840	237	1334	28.21% $\pm$ 3.04%	17.77% $\pm$ 2.05%	21.80%	20	8.44%
Googlec.+Wikic.	instanceOf	LEILA(SVM)	563	203	1334	36.06% $\pm$ 3.97%	15.22% $\pm$ 1.93%	21.40%	11	5.42%
Googlec.+Wikic.	instanceOf	LEILA(kNN)	826	246	1334	29.78% $\pm$ 3.12%	18.44% $\pm$ 2.08%	22.78%	19	7.72%

#O – number of output pairs
#C – number of correct output pairs
#I – number of ideal pairs
#E – number of examples among #C
%E – proportion of examples among #C
Recall and Precision with confidence interval at $\alpha = 95\%$

4.2.2 Results with different competitors

Table 2 shows the results for comparing LEILAagainst various competitors (with LEILAperformance in boldface). The numbers show that LEILAclearly outperformed the other approaches in almost all cases. We compared LEILAto **TextToOnto** and **Text2Onto** for the `instanceOf` relation on the Wikicomposers corpus. TextToOnto requires an ontology as source of possible concepts. We gave it the WordNet ontology, so that it had the same preconditions as LEILA.

TextToOnto does not have any tuning parameters. Text2Onto allows the choice of certain sub-algorithms. The reported results were the best we could achieve. Text2Onto seems to have a precision comparable to ours, although the small number of found pairs does not allow a significant conclusion. Both systems have drastically lower recall than LEILA.

Next, we compared LEILA to **Snowball**. As described above, we used the `company/headquarters`-relation and the corpus that came with Snowball. Since we only had access to the test corpus, we trained LEILA on a small portion (3%) of the test documents and tested on the remaining ones. Since the original 5 seed pairs that Snowball used did not appear in the collection at our disposal, we chose 5 other seed pairs as examples. We used no counterexamples and hence omitted the learning step of our algorithm.

LEILA quickly finds the pattern "Y-based X". This led to very high precision and good recall, compared to Snowball – even though Snowball was trained on a much larger training collection of some thousand documents. In the original paper [1], Snowball is evaluated using the *Ideal Metric*. Consequently, we report precision and recall with respect to the Ideal Metric. By the nature of this metric, the precision increases and the recall decreases, although the relative performance of the systems does not change.

Finally, we compared LEILA to the **CV-system**. In the gold standard for this approach, the ideal pairs are given as a table, in which each entity is assigned to its most likely concept according to a human understanding of the text, independently of whether there are explicit definitions for the entity in the text or not. Some caution is necessary in a comparison. For example, the text might state that Leonardo Da Vinci was a painter, but that he also invented new machines. Then our system will classify him as a painter, whereas the competitor might say he is an inventor. We demonstrate the difference of the two approaches by our experiments.

We conducted two experiments: First, we used the document set used in Cimiano and Völker's original paper [11], the Lonely Planet corpus. To ensure a fair comparison, we trained LEILA separately on the Wikicomposers corpus, so that LEILA cannot have example pairs in its output. For the evaluation, we calculated precision and recall with respect to an ideal table provided by the authors. Since our competitor uses a different ontology, we allowed a distance of 4 edges in the WordNet hierarchy to count as a match (for both systems). 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 competitor run on the Wikicomposers corpus. As the CV-system requires a set of target concepts, we gave it the set of all concepts in our ideal pairs. Furthermore, the system requires an ontology on these concepts. We gave it the WordNet ontology, pruned to the target concepts with

their super-concepts. We evaluated by the Relaxed Ideal Metric, again allowing a distance of 4 edges in the WordNet hierarchy to count as a match (for both systems). This time, our competitor performed worse. This is because our ideal table is constructed from the definitions in the text, which our competitor is not designed to follow. These experiments show the different philosophies of the CV-system and LEILA.

Table 4.2: Results with different competitors

Corpus	M	Relation	System	#O	#C	#I	Precision		Recall		F1
Snowball corp.	S	headquarters	LEILA(SVM)	92	82	165	**89.13%**	± 6.36%	**49.70%**	± 7.63%	**63.81%**
Snowball corp.	S	headquarters	LEILA(kNN)	91	82	165	**90.11%**	± 6.13%	**49.70%**	± 7.63%	**64.06%**
Snowball corp.	S	headquarters	Snowball	144	49	165	34.03%	± 7.74%	29.70%	± 6.97%	31.72%
Snowball corp.	I	headquarters	LEILA(SVM)	50	48	126	**96.00%**	± 5.43%	**38.10%**	± 8.48%	**54.55%**
Snowball corp.	I	headquarters	LEILA(kNN)	49	48	126	**97.96%**	± 3.96%	**38.10%**	± 8.48%	**54.86%**
Snowball corp.	I	headquarters	Snowball	64	31	126	48.44%	±12.24%	24.60%	± 7.52%	32.63%
Wikicomposers	S	instanceOf	LEILA(SVM)	685	408	1127	**59.56%**	± 3.68%	**36.20%**	± 2.81%	**45.03%**
Wikicomposers	S	instanceOf	LEILA(kNN)	790	463	1127	**58.61%**	± 3.43%	**41.08%**	± 2.87%	**48.30%**
Wikicomposers	S	instanceOf	Text2Onto	36	18	1127	50.00%		1.60%	± 0.73%	3.10%
Wikicomposers	S	instanceOf	TextToOnto	121	47	1127	38.84%	± 8.68%	4.17%	± 1.17%	7.53%
Wikicomposers	R	instanceOf	LEILA(SVM)	336	257	744	**76.49%**	± 4.53%	**34.54%**	± 3.42%	**47.59%**
Wikicomposers	R	instanceOf	LEILA(kNN)	367	276	744	**75.20%**	± 4.42%	**37.10%**	± 3.47%	**49.68%**
Wikicomposers	R	instanceOf	CV-system	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 document-based, I: Ideal Matric, R: Relaxed Ideal Metric)
Other abbreviations as in Table 4.1

5 Conclusion and Outlook

We have presented a novel approach for extracting binary relations from natural language text. The linguistic and statistical techniques we employ provide us with patterns that are robust to variations in the text. We have implemented our approach and showed that our system LEILA outperforms existing competitors.

Our current implementation leaves room for future work. First, the linkages allow for more sophisticated ways of resolving anaphoras and other references as compared to what we have currently implemented. Better methods could be used to detect named entities. Furthermore, patterns could be matched in a more flexible way, for example by allowing matches of semantically similar words in the bridge (like "to recognize as" and "to know as"). Also, the patterns could be learned and optimized only once and then be applied to different corpora.

The system could learn numerous interesting relations, including for example `country/president`, `partOf`, `belongsTo`, `isAuthorOf` or `isMarriedTo`. If only the results with a high confidence are taken, the system could be used for building an ontology automatically. The system could acquire and exploit new corpora on its own (for example, it could read newspapers). Once it has built up sufficient background knowledge, it could use this knowledge to analyze corpora more efficiently. For example, once it knows that "Rita" was a hurricane, it will conclude that "Rita (2005)" does not mean that Rita was born in 2005, but rather that Rita occurred in 2005. The system could make more sophisticated use of the semantic relations, e.g. by concluding that multiple inheritance in the taxonomy might indicate polysemy. We plan to exploit these possibilities in our future work.

Appendix A Proof for the bound in section 2.3.1

With the definitions given in section 2.3.1, we prove an upper bound for the probability $P(\#A > \#B)$. Let $\#C = N - \#A - \#B$.

$$\begin{aligned} P(\#A > \#B) &= P(\#A > \#B | \#C \geq k) \cdot P(\#C \geq k) \\ &\quad + P(\#A > \#B | \#C < k) \cdot P(\#C < k) \quad \text{for any } k \end{aligned}$$

$$\begin{aligned} &\leq P(\#C \geq k) + P(\#A > \#B | \#C < k) \cdot P(\#C < k) \\ &= P(\#C \geq k) + \sum_{i=0}^{k-1} P(\#A > \frac{N-i}{2} | \#C = i) \cdot P(\#C = i) \\ &\leq P(\#C \geq k) + \max_{i=0\ldots k-1} P(\#A > \frac{N-i}{2} | \#C = i) \cdot \sum_{i=0}^{k-1} P(\#C = i) \\ &\leq P(\#C \geq k) + \max_{i=0\ldots k-1} P(\#A > \frac{N-i}{2} | \#C = i) \end{aligned}$$

Now, we make use of the Chernoff-Hoeffding bound. This bound can be generalized for any $x > pn$:

$$P(\sum_{i=0}^{n} X_i \geq x) \quad \leq \quad 2e^{-2n(\frac{x}{n}-p)^2}$$

If we choose $k = (1+\varepsilon)p_C N$, we may write:

$$P(\#C \geq k) \leq 2 \cdot \exp^{-2N \cdot (\frac{k}{N} - p_C)^2} .$$

We observe that the moment we fix the $\#C$, we obtain a Binomial distribution for A and B with the parameter

$\tilde{p_A} = p_A/(p_A + p_B)$. Since $p_A < p_B$, $\tilde{p_A} < \frac{1}{2}$ and $(N-i)/2 > \tilde{p_A}(N-i)$. Therefore

$$P(\#A > \frac{N-i}{2} | \#C = i) \leq 2e^{-2(N-i)(\frac{1}{2}-\tilde{p_A})^2}$$

Using these two bounds yields:

$$\begin{aligned} P(\#A > \#B) &\leq 2e^{-2N(\frac{k}{N}-p_C)^2} + \max_{i=0\ldots k-1} 2e^{-2(N-i)(\frac{1}{2}-\tilde{p_A})^2} \\ &= 2e^{-2N(\frac{k}{N}-p_C)^2} + 2e^{-2(N-(k-1))(\frac{1}{2}-\tilde{p_A})^2} \\ &= 2e^{-2N\varepsilon^2 p_C^2} + 2e^{-2[N(1-p_C-\varepsilon p_C)+1](\frac{1}{2}-\tilde{p_A})^2} \end{aligned}$$

We choose $\varepsilon = \frac{1-p_C}{2p_C}$. Then

$$P(\#A > \#B) \leq 2e^{-2N(\frac{1-p_C}{2p_C})^2 p_C^2} + 2e^{-2[N(1-p_C-\frac{1-p_C}{2p_C}p_C)+1](\frac{1}{2}-\tilde{p_A})^2}$$

$$\begin{aligned} &= 2e^{-2N\frac{(1-p_C)^2}{4}} + 2e^{-2[N(1-p_C-\frac{1-p_C}{2})+1](\frac{1}{2}-\tilde{p_A})^2} \\ &= 2e^{-N\frac{(1-p_C)^2}{2}} + 2e^{-2[N(\frac{1-p_C}{2})+1](\frac{1}{2}-\tilde{p_A})^2} \\ &= 2e^{-N\frac{(1-p_C)^2}{2}} + 2e^{-(N(1-p_C)+2)(\frac{1}{2}-\tilde{p_A})^2} \end{aligned}$$

Bibliography

[1] E. Agichtein and L. Gravano. *Snowball*: extracting relations from large plain-text collections. In *Proc. of the 5th ACM Conference on Digital Libraries*, 2000.

[2] F. Alexander. A parser for HPSG. *Technical report, Carnegie Mellon University*, 1990.

[3] J. Aslam and S. Decatur. On the sample complexity of noise-tolerant learning. *Information Processing Letters*, 57(4):189–195, 1996.

[4] S. Brin. Extracting patterns and relations from the world wide web. In *Selected papers from the International Workshop on WWW*, 1999.

[5] P. Buitelaar, D. Olejnik, and M. Sintek. A protege plug-in for ontology extraction from text based on linguistic analysis. In *Proc. of ESWS-04*, 2004.

[6] P. Buitelaar and S. Ramaka. Unsupervised ontology-based semantic tagging for knowledge markup. In W. Buntine, A. Hotho, and S. Bloehdorn, editors, *Proc. of the LWS Workshop at ICML*, 2005.

[7] M. J. Cafarella, D. Downey, S. Soderland, and O. Etzioni. Knowitnow: Fast, scalable information extraction from the web. In *Proc. of HLT/EMNLP*, pages 563–570, 2005.

[8] M. Califf and R. Mooney. Relational learning of pattern-match rules for information extraction. *Proc. of NLL Workshop at ACL*, pages 9–15, 1997.

[9] V. Cherkassky and M. Yunqian. Practical selection of svm parameters and noise estimation for svm regression. *Neural Networks*, 17(1):113–126, 2004.

[10] P. Cimiano, G. Ladwig, and S. Staab. Gimme the context: Contextdriven automatic semantic annotation with cpankow. In A. Ellis and T. Hagino, editors, *Proc. of the 14th WWW*, Chiba, Japan, 2005.

[11] P. Cimiano and J. Volker. Text2onto - a framework for ontology learning and data-driven change discovery. In A. Montoyo, R. Munozand, and E. Metais, editors, *Proc. of the 10th NLDB*, 2005.

[12] P. Cimiano and J. Volker. Towards large-scale, open-domain and ontology-based named entity classification. In *Proc. of the RANLP*, 2005.

[13] M. Collins. Three generative, lexicalised models for statistical parsing. In *Proc. of the 35th ACL*, 1997.

[14] O. Etzioni, M. Cafarella, D. Downey, S. Kok, A. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates. Web-scale information extraction in knowitall (preliminary results). In *Proc. of the 13th WWW*, pages 100–110, 2004.

[15] C. Fellbaum. *WordNet: An Electronic Lexical Database*. MIT Press, 1998.

[16] A. Finn and N. Kushmerick. Multi-level boundary classification for information extraction. In *ECML*, pages 111–122, 2004.

[17] D. Freitag and N. Kushmerick. Boosted wrapper induction. In *Proc. of ANCAI*, 2000.

[18] J. Graupmann. Concept-based search on semi-structured data exploiting mined semantic relations. In *EDBT Workshops*, pages 34–43, 2004.

[19] A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proc. of the 14th ICCL*, 1992.

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

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

[22] M. J. Kearns and R. E. Schapire. Efficient distribution-free learning of probabilistic concepts. In S. J. Hanson, G. A. Drastal, and R. L. Rivest, editors, *COLT*. Bradford/MIT Press, 1994.

[23] D. Klein and C. D. Manning. Fast exact inference with a factored model for natural language parsing. In *Proc of 15th NIPS*, 2002.

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

[25] A. Maedche and S. Staab. Discovering conceptual relations from text. In W. Horn, editor, *Proc. of the 14th ECAI*, 2000.

[26] C. D. Manning and H. Schutze. *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.

[27] J. Maxwell and R. Kaplan. An efficient parser for LFG. In *Proc. of the 1st LFG*, 1996.

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

[29] I. Niles and A. Pease. Towards a Standard Upper Ontology. In C. Welty and B. Smith, editors, *Proc. of the 2nd IFOIS*, 2001.

[30] D. Ravichandran and E. Hovy. Learning surface text patterns for a question answering system. In *Proc. of the 40th ACL*, 2002.

[31] E. Riloff. Automatically generating extraction patterns from untagged text. *Proc. of the 13th ACAI*, pages 1044–1049, 1996.

[32] E. Rosch, C. Mervis, W. Gray, D. Johnson, and P. Boyes-Bream. Basic objects in natural categories. *Cognitive Psychology*, pages 382–439, 1976.

[33] H. U. Simon. General bounds on the number of examples needed for learning probabilistic concepts. In *Proc. of the 6th COLT*, 1993. ACM Press.

[34] D. Sleator and D. Temperley. Parsing english with a link grammar. *Proc. of 3rd IWPT*, 1993.

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

[36] S. Soderland, D. Fisher, J. Aseltine, and W. Lehnert. Crystal: Inducing a conceptual dictionary. *Proc. of the 14th IJCAI*, pages 1314–1319, 1995.

[37] F. Xu and H. U. Krieger. Integrating shallow and deep nlp for information extraction. In *Proc. of RANLP*, 2003.

[38] F. Xu, D. Kurz, J. Piskorski, and S. Schmeier. Term extraction and mining term relations from free-text documents in the financial domain. In *Proc. of the 5th BIS*, 2002.

[39] R. Yangarber, R. Grishman, P. Tapanainen, and S. Huttunen. Automatic acquisition of domain knowledge for information extraction. In *Proc. of the 18th ICCL*, 2000. ACL.

[40] R. Yangarber, W. Lin, and R. Grishman. Unsupervised learning of generalized names. In *Proc. of the 19th ICCL*, 2002. ACL.

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

Below you find a list of the most recent technical reports of the Max-Planck-Institut für Informatik. They are available by anonymous ftp from `ftp.mpi-sb.mpg.de` under the directory `pub/papers/reports`. Most of the reports are also accessible via WWW using the URL `http://www.mpi-sb.mpg.de`. If you have any questions concerning ftp or WWW access, please contact `reports@mpi-sb.mpg.de`. Paper copies (which are not necessarily free of charge) can be ordered either by regular mail or by e-mail at the address below.

Max-Planck-Institut für Informatik
Library
attn. Anja Becker
Stuhlsatzenhausweg 85
66123 Saarbrücken
GERMANY
e-mail: `library@mpi-sb.mpg.de`

MPI-I-2006-5-004	F. Suchanek	Combining Linguistic and Statistical Analysis to Extract Relations from web Documents
MPI-I-2006-5-003	V. Scholz, M. Magnor	Garment Texture Editing in Monocular Video Sequences based on Color-Coded Printing Patterns
MPI-I-2006-5-002	H. Bast, D. Majumdar, R. Schenkel, M. Theobald, G. Weikum	IO-Top-k: Index-access Optimized Top-k Query Processing
MPI-I-2006-5-001	M. Bender, S. Michel, G. Weikum, P. Triantafilou	Overlap-Aware Global df Estimation in Distributed Information Retrieval Systems
MPI-I-2006-1-001	M. Dimitrios	?
MPI-I-2005-5-002	S. Siersdorfer, G. Weikum	Automated Retraining Methods for Document Classification and their Parameter Tuning
MPI-I-2005-4-006	C. Fuchs, M. Goesele, T. Chen, H. Seidel	An Emperical Model for Heterogeneous Translucent Objects
MPI-I-2005-4-005	G. Krawczyk, M. Goesele, H. Seidel	Photometric Calibration of High Dynamic Range Cameras
MPI-I-2005-4-004	C. Theobalt, N. Ahmed, E. De Aguiar, G. Ziegler, H. Lensch, M.A.,. Magnor, H. Seidel	Joint Motion and Reflectance Capture for Creating Relightable 3D Videos
MPI-I-2005-4-003	T. Langer, A.G. Belyaev, H. Seidel	Analysis and Design of Discrete Normals and Curvatures
MPI-I-2005-4-002	O. Schall, A. Belyaev, H. Seidel	Sparse Meshing of Uncertain and Noisy Surface Scattered Data
MPI-I-2005-4-001	M. Fuchs, V. Blanz, H. Lensch, H. Seidel	Reflectance from Images: A Model-Based Approach for Human Faces
MPI-I-2005-2-004	Y. Kazakov	A Framework of Refutational Theorem Proving for Saturation-Based Decision Procedures
MPI-I-2005-2-003	H.d. Nivelle	Using Resolution as a Decision Procedure
MPI-I-2005-2-002	P. Maier, W. Charatonik, L. Georgieva	Bounded Model Checking of Pointer Programs
MPI-I-2005-2-001	J. Hoffmann, C. Gomes, B. Selman	Bottleneck Behavior in CNF Formulas
MPI-I-2005-1-008	C. Gotsman, K. Kaligosi, K. Mehlhorn, D. Michail, E. Pyrga	Cycle Bases of Graphs and Sampled Manifolds
MPI-I-2005-1-008	D. Michail	?
MPI-I-2005-1-007	I. Katriel, M. Kutz	A Faster Algorithm for Computing a Longest Common Increasing Subsequence
MPI-I-2005-1-003	S. Baswana, K. Telikepalli	Improved Algorithms for All-Pairs Approximate Shortest Paths in Weighted Graphs
MPI-I-2005-1-002	I. Katriel, M. Kutz, M. Skutella	Reachability Substitutes for Planar Digraphs
MPI-I-2005-1-001	D. Michail	Rank-Maximal through Maximum Weight Matchings
MPI-I-2004-NWG3-001	M. Magnor	Axisymmetric Reconstruction and 3D Visualization of Bipolar Planetary Nebulae
MPI-I-2004-NWG1-001	B. Blanchet	Automatic Proof of Strong Secrecy for Security Protocols
MPI-I-2004-5-001	S. Siersdorfer, S. Sizov, G. Weikum	Goal-oriented Methods and Meta Methods for Document Classification and their Parameter Tuning
MPI-I-2004-4-006	K. Dmitriev, V. Havran, H. Seidel	Faster Ray Tracing with SIMD Shaft Culling

MPI-I-2004-4-005	I.P. Ivrissimtzis, W.-. Jeong, S. Lee, Y.a. Lee, H.-. Seidel	Neural Meshes: Surface Reconstruction with a Learning Algorithm
MPI-I-2004-4-004	R. Zayer, C. Rssl, H. Seidel	r-Adaptive Parameterization of Surfaces
MPI-I-2004-4-003	Y. Ohtake, A. Belyaev, H. Seidel	3D Scattered Data Interpolation and Approximation with Multilevel Compactly Supported RBFs
MPI-I-2004-4-002	Y. Ohtake, A. Belyaev, H. Seidel	Quadric-Based Mesh Reconstruction from Scattered Data
MPI-I-2004-4-001	J. Haber, C. Schmitt, M. Koster, H. Seidel	Modeling Hair using a Wisp Hair Model
MPI-I-2004-2-007	S. Wagner	Summaries for While Programs with Recursion
MPI-I-2004-2-002	P. Maier	Intuitionistic LTL and a New Characterization of Safety and Liveness
MPI-I-2004-2-001	H. de Nivelle, Y. Kazakov	Resolution Decision Procedures for the Guarded Fragment with Transitive Guards
MPI-I-2004-1-006	L.S. Chandran, N. Sivadasan	On the Hadwiger's Conjecture for Graph Products
MPI-I-2004-1-005	S. Schmitt, L. Fousse	A comparison of polynomial evaluation schemes
MPI-I-2004-1-004	N. Sivadasan, P. Sanders, M. Skutella	Online Scheduling with Bounded Migration
MPI-I-2004-1-003	I. Katriel	On Algorithms for Online Topological Ordering and Sorting
MPI-I-2004-1-002	P. Sanders, S. Pettie	A Simpler Linear Time 2/3 - epsilon Approximation for Maximum Weight Matching
MPI-I-2004-1-001	N. Beldiceanu, I. Katriel, S. Thiel	Filtering algorithms for the Same and UsedBy constraints
MPI-I-2003-NWG2-002	F. Eisenbrand	Fast integer programming in fixed dimension
MPI-I-2003-NWG2-001	L.S. Chandran, C.R. Subramanian	Girth and Treewidth
MPI-I-2003-4-009	N. Zakaria	FaceSketch: An Interface for Sketching and Coloring Cartoon Faces
MPI-I-2003-4-008	C. Roessl, I. Ivrissimtzis, H. Seidel	Tree-based triangle mesh connectivity encoding
MPI-I-2003-4-007	I. Ivrissimtzis, W. Jeong, H. Seidel	Neural Meshes: Statistical Learning Methods in Surface Reconstruction
MPI-I-2003-4-006	C. Roessl, F. Zeilfelder, G. Nrnberger, H. Seidel	Visualization of Volume Data with Quadratic Super Splines
MPI-I-2003-4-005	T. Hangelbroek, G. Nrnberger, C. Roessl, H.S. Seidel, F. Zeilfelder	The Dimension of C^1 Splines of Arbitrary Degree on a Tetrahedral Partition
MPI-I-2003-4-004	P. Bekaert, P. Slusallek, R. Cools, V. Havran, H. Seidel	A custom designed density estimation method for light transport
MPI-I-2003-4-003	R. Zayer, C. Roessl, H. Seidel	Convex Boundary Angle Based Flattening
MPI-I-2003-4-002	C. Theobalt, M. Li, M. Magnor, H. Seidel	A Flexible and Versatile Studio for Synchronized Multi-view Video Recording
MPI-I-2003-4-001	M. Tarini, H.P.A. Lensch, M. Goesele, H. Seidel	3D Acquisition of Mirroring Objects
MPI-I-2003-2-004	A. Podelski, A. Rybalchenko	Software Model Checking of Liveness Properties via Transition Invariants
MPI-I-2003-2-003	Y. Kazakov, H. de Nivelle	Subsumption of concepts in DL $\mathcal{FL}_0$ for (cyclic) terminologies with respect to descriptive semantics is PSPACE-complete
MPI-I-2003-2-002	M. Jaeger	A Representation Theorem and Applications to Measure Selection and Noninformative Priors
MPI-I-2003-2-001	P. Maier	Compositional Circular Assume-Guarantee Rules Cannot Be Sound And Complete