SOFIE: A Self-Organizing Framework for Information Extraction

Fabian M. Suchanek
Max-Planck Institute CS
Saarbruecken, Germany
suchanek@mpii.de

Mauro Sozio
Max-Planck Institute CS
Saarbruecken, Germany
msozio@mpii.de

Gerhard Weikum
Max-Planck Institute CS
Saarbruecken, Germany
weikum@mpii.de

ABSTRACT

This paper presents SOFIE, a system for automated ontology extension. SOFIE can parse natural language documents, extract ontological facts from them and link the facts into an ontology. SOFIE uses logical reasoning on the existing knowledge and on the new knowledge in order to disambiguate words to their most probable meaning, to reason on the meaning of text patterns and to take into account world knowledge axioms. This allows SOFIE to check the plausibility of hypotheses and to avoid inconsistencies with the ontology. The framework of SOFIE unites the paradigms of pattern matching, word sense disambiguation and ontological reasoning in one unified model. Our experiments show that SOFIE delivers high-quality output, even from unstructured Internet documents.

Categories and Subject Descriptors

H.1. [**Information Systems**]: Models and Principles

General Terms

Algorithms, Design

Keywords

Ontology, Information Extraction, Automated Reasoning

1. INTRODUCTION

1.1 Background and Motivation

Recently, several projects such as YAGO [37, 38], Kylin/KOG [42, 43], and DBpedia [4], have successfully constructed large ontologies by using Information Extraction (IE).[1] These ontologies contain millions of entities and tens of millions of facts (i.e., instances of relations between entities). A hierarchy of classes gives them a clean taxonomic structure. Empirical assessment has shown that these approaches can achieve an accuracy of over 95 percent.

The focus in these projects has been on extracting information from the semi-structured components of Wikipedia (such as infoboxes and the category system). In order to achieve an even broader coverage, new sources must be brought into scope. One particularly rich source are natural-language documents, such as news articles, biographies, scientific publications, and also the full text of Wikipedia articles. But so far even the best IE methods have typically achieved only 80 percent accuracy (or less) in such settings. While this may be good enough for some applications, the error rate is unacceptable for an ontological knowledge base. The key idea to overcome this dilemma, pursued in this paper, is to leverage the existing ontology for its own growth. We propose to use trusted facts as a basis for generating good text patterns. These patterns guide the IE from natural language text. The resulting new hypotheses are scrutinized with regard to their consistency with the already known facts. This will allow extracting ontological facts of high quality even from unstructured text documents.

1.2 Example Scenario

Assume that a knowledge-gathering system encounters the following sentence:

Einstein attended secondary school in Germany.

Knowing that *"Einstein"* is the family name of Albert Einstein and knowing that Albert Einstein was born in Germany, the system might deduce that "X attended secondary school in Y" is a good indicator of X being born in Y. Now imagine the system finds the sentence

Elvis attended secondary school in Memphis.

Many people have called themselves *"Elvis"*. In the present case, assume that the context indicates that Elvis Presley is meant. But the system already knows (from the facts it has already gathered) that Elvis Presley was born in the State of Mississippi. Knowing that a person can only be born in a single location and knowing that Memphis is not located in Mississippi, the system concludes that the pattern "X attended secondary school in Y" cannot mean that X was born in Y. Re-considering the first sentence, it finds that *"Einstein"* could have meant Hermann Einstein instead. Hermann was the father of Albert Einstein. Knowing that Hermann went to school in Germany, the system figures out that the pattern "X attended secondary school in Y" rather indicates that someone went to school in some place. Therefore, the system deduces that Elvis went to school in Memphis.[2] This is the kind of "intelligent" IE that we pursue in this paper.

1.3 Contribution

The example scenario shows that extracting new facts that are consistent with an existing ontology entails several, highly intertwined problems:

Pattern selection: Facts are extracted from natural language documents by finding meaningful patterns in the text.

[1] In this paper, *ontology* means a knowledge base of semantic facts.

WWW 2009, April 20–24, 2009, Madrid, Spain.
ACM 978-1-60558-487-4/09/04.

[2] This is actually true. Albert Einstein went to secondary school in Switzerland, not Germany.

The accuracy of this technique critically depends on having a variety of meaningful patterns. It can be further boosted if counter-productive patterns are excluded systematically [36]. Thus, discovering and assessing patterns is a key task of IE.

Entity disambiguation: For ontological IE, the words or phrases from the text have to be mapped to entities in the ontology. In many cases, this mapping is ambiguous. The word "Paris", e.g., can denote either the French capital or a city in Texas. Since many location names, companies, or product names are ambiguous, finding the intended meaning of a word is often a difficult task.

Consistency checking: The newly extracted facts have to be logically consistent with the existing ontology. Consistency checking is an interesting problem by itself. In our case, the problem is particularly challenging, because a large set of IE-provided noisy candidates has to be scrutinized against a trusted core of facts.

This paper presents a new approach to these problems. Rather than addressing each of them separately, we provide a unified model for ontology-oriented IE that solves all three issues simultaneously. To this end, we cast known facts, hypotheses for new facts, word-to-entity mappings, gathered sets of patterns, and a configurable set of semantic constraints into a unified framework of logical clauses. Then, all three problems together can be seen as a Weighted MAX-SAT problem, i.e., as the task of identifying a maximal set of consistent clauses. The approach is fully implemented in a system for knowledge gathering and ontology maintenance, coined SOFIE. The salient properties of SOFIE and novel research contributions of this paper are the following:

- a new model for consistent growth of a large ontology;
- a unified method for pattern selection, entity disambiguation, and consistency checking;
- an efficient algorithm for the resulting Weighted MAX-SAT problem that is tailored to the specific task of ontology-centric IE;
- experiments with a variety of real-life textual and semi-structured sources to demonstrate the scalability and high accuracy of the approach.

The rest of the paper is organized as follows. Section 2 discusses related work, Sections 3 and 4 present the SOFIE model and its implementation, and Section 5 discusses experiments.

2. RELATED WORK

Fact Gathering. Unlike manual approaches such as WordNet [18], Cyc [23] or SUMO [26], IE approaches seek to extract facts from text documents automatically. They encompass a wide variety of models and methods, including linguistic, learning, and rule-based approaches [32]. The methods often start with a given set of target relations and aim to collect as many of their instances (the facts) as possible. These facts can serve for the purposes of ontology population or ontology learning.

DIPRE [10], Snowball [2] and KnowItAll [17] are among the most prominent projects of this kind. They harness manually specified seed facts of a given relation (e.g., a small number of company-city pairs for a headquarter relation) to find textual patterns that could possibly express the relation, use statistics to identify the best patterns, and then find new facts from occurrences of these patterns. LEILA [36] has further improved this method by using both examples and counterexamples as seeds, in order to generate more robust patterns. This notion of counterexamples is also adopted by SOFIE. Blohm et al. [9, 8] provide enhanced methods for selecting the best patterns.

TextRunner [5] pursues the even more ambitious goal of extracting all instances of *all* meaningful relations from Web pages, a paradigm referred to as *Open IE* [16]. However, all of these projects extract merely *non-canonical facts*. This means (1) that they do not disambiguate words to entities and (2) that they do not extract well-defined relations (but, e.g., verbal phrases). In contrast, SOFIE delivers canonicalized output that can be directly used in a formal ontology.

Wikipedia-centric Approaches. Recently, a number of projects have applied IE with specific focus on Wikipedia: DBpedia [4], work by Ponzetto et. al. [27], Kylin/KOG [42, 43], and our own YAGO project [37, 38]. While Ponzetto et al. focus on extracting a taxonomic hierarchy from Wikipedia, DBpedia and YAGO construct full-fledged ontologies from the semi-structured parts of Wikipedia (i.e., from infoboxes and the category system). SOFIE, on the other hand, can process the full body of Wikipedia articles. It is not even tied to Wikipedia but can handle arbitrary Web pages and natural-language texts.

Kylin goes beyond the IE in DBpedia and YAGO by extracting information not just from the infoboxes and categories, but also from the full text of the Wikipedia articles. KOG (Kylin Ontology Generator) builds on Kylin's output, unifies different attribute names, derives type signatures, and (like YAGO) maps the entities onto the WordNet taxonomy, using Markov Logic Networks [31]. KOG builds on the class system of YAGO and DBpedia (along with the entities in each class) to generate a taxonomy of classes. Both Kylin and KOG are customized and optimized for Wikipedia articles, while this paper aims at IE from arbitrary Web sources.

Wang et al [41] have presented an approach called *Positive-Only Relation Extraction* (PORE). PORE is a holistic pattern matching approach, which has been implemented for relation-instance extraction from Wikipedia. Unlike the approach presented in this paper, PORE does not incorporate world knowledge, which would be necessary for ontology building and extension.

Declarative IE. Shen et al. [33] propose a framework for declarative IE, based on Datalog. By encapsulating the non-declarative code into predicates, the framework provides a clean model for rule-based information extraction and allows consistency constraints and checks against existing facts (e.g., for entity resolution). The approach has been successfully applied for building and maintaining community portals like DBlife [15], while the universal ontologies studied in this paper are not in the scope of the work.

Reiss et al. [30] pursue a declarative approach that is similar to that of [33], but use database-style algebraic operator trees rather than Datalog. The approach greatly simplifies the manageability of large-scale IE tasks, but does not address any ontology-centered issues.

Poon et al. [28] use Markov Logic networks [31] for IE. Their approach can simultaneously tokenize bibliographic entries and reconcile the extracted entities. In Markov Logic, first-order formulas that express properties of patterns and

hypotheses are grounded and translated into a Markov random field that defines a clique-factorized joint probability distribution for the entirety of hypotheses. Inferencing procedures over such structures can compute probabilities for the truth of the various hypotheses. Our approach has algorithmic building blocks in common with [28], but follows a very different architectural paradigm. Rather than performing probabilistic inferences on the extracted entities, our approach aims to identify the best subset of hypotheses that is consistent with the textual patterns, the existing ontology and the semantic constraints.

Ontology Integration and Extension. The goal of the current paper is to provide means for automatically extending an ontology by new facts found by IE methods – while preserving the ontology's consistency. This setting resembles the issue of ontology integration: merging two or more ontologies in a consistent manner [34, 40]. However, our setting is much more difficult, because the new facts are extracted from highly noisy text and Web sources rather than from a second, already formalized and clean, ontology.

Boer et. al. [13] present an approach for extending a given ontology, based on a co-occurrence analysis of entities in Web documents. However, they rely on the existence of documents that list instances of a certain relation. While these documents exist for some relations, they do not exist for many others; this limits the applicability of the approach.

Banko et al. pursue a similar goal called *Lifelong Learning* [6], implemented in the ALICE system. ALICE is based on a core ontology and aims to extend it by new facts using statistical methods. The approach has not been tried out with individual entities (in canonicalized form). Moreover, it lacks logical reasoning capabilities that are crucial for ensuring the consistency of the automatically extended ontology.

We believe that SOFIE is the very first approach to the ontology-extension problem that integrates logical constraint checking with pattern-based IE, and is thus able to provide ontological facts about disambiguated entities in canonical form.

3. MODEL

3.1 Statements

Facts and Hypotheses. A *statement* is a tuple of a relation and a pair of entities. A statement has an associated *truth value* of 1 or 0. We denote the truth value of a statement in brackets:

bornIn(*AlbertEinstein*, *Ulm*)[1]

A statement with truth value 1 is called a *fact*. A statement with an unknown truth value is called a *hypothesis*.

Ontological Facts. SOFIE is designed to extend an existing ontology. In principle, SOFIE could work with any ontology that can be expressed as a set of facts. For our experiments, we used the YAGO ontology [37], which can be expressed as follows:

type(*AlbertEinstein*, *Physicist*)[1]
subclassOf(*Physicist*, *Scientist*)[1]
bornIn(*AlbertEinstein*, *Ulm*)[1]
...

Wics. As a knowledge gathering system, SOFIE has to address the problem that most words have several meanings. The word "*Java*", for example, can refer to the programming language or to the Indonesian island.[3] In a given context, however, a word is very likely to have only one meaning [19]. We define a *word in context (wic)* as a pair of a word and a context. For us, the context of a word simply is the document in which the word appears. Thus, a wic is a pair of a word and a document identifier.[4] We use the notation *word@doc*, so that, e.g., the word "*Java*" in the document *D8* is denoted by *Java@D8*. We assume that all occurrences of a wic have the same meaning.

Textual Facts. SOFIE has a component for extracting surface information from a given text corpus. This information also takes the form of facts. One type of facts makes assertions about the occurrences of patterns. For example, the system might find that the pattern "X went to school in Y" occurred with the wics *Einstein@D29* and *Germany@D29*. This results in the following fact:

patternOcc("X went to school in Y",
Einstein@D29, *Germany@D29*)[1]

Another type of facts states how likely it is, from a linguistic point of view, that a wic refers to a certain entity. We call this likeliness value the *disambiguation prior*. One way of computing a disambiguation prior is to exploit context statistics (see Section 4). Here, we just give an example for facts about the disambiguation prior of the wic *Elvis@D29*:

disambPrior(*Elvis@D29*, *ElvisPresley*, 0.8)[1]
disambPrior(*Elvis@D29*, *ElvisCostello*, 0.2)[1]

Hypotheses. Based on the ontological facts and the textual facts, SOFIE forms hypotheses. These hypotheses can concern the disambiguation of wics. For example, SOFIE can hypothesize that *Java@D8* should be disambiguated as the programming language Java:

disambiguateAs(*Java@D8*, *JavaProgrammingLanguage*)[?]

We use a question mark to indicate the unknown truth value of the statement. SOFIE can also hypothesize about whether a certain pattern expresses a certain relation:

expresses("X was born in Y", *bornInLocation*)[?]

Apart from textual hypotheses, SOFIE also forms hypotheses about potential new facts. For example, SOFIE could establish the hypothesis that Java was developed by Microsoft:

developed(*Microsoft*, *JavaProgrammingLanguage*)[?]

Unified Model. By casting both the ontology and the linguistic analysis of the documents into statements, SOFIE unifies ontology-based reasoning and information extraction: everything takes the form of statements. SOFIE will aim to figure out which hypotheses are likely to be true. For this purpose, SOFIE uses *rules*.

3.2 Rules

Literals and Rules. SOFIE will use logical knowledge to figure out which hypotheses are likely to be true. This knowledge takes the from of *rules*. Rules are based on *literals*. A literal is a statement that can have placeholders for

[3]This is the problem of polysemy, where one word refers to multiple entities. Conversely, if one entity has multiple names (synonymy) this does not pose a problem, as SOFIE maps words to entities.

[4]A wic is related to but different from a *concordance* aka. KWIC (keyword in context) [24].

the relation or some of the entities, e.g., *bornIn*(*X*, *Ulm*). A *rule* is a propositional first order logic formula over literals, e.g., *bornIn*(*X*, *Ulm*) ⇒ ¬ *bornIn*(*X*, *Timbuktu*). As in Prolog and Datalog, all placeholders are implicitly universally quantified. We postpone the discussion of the formal semantics of the rules to Section 3.3 and stay with an intuitive understanding of rules for the moment.

Grounding. A *ground instance of a literal* is a statement obtained by replacing the placeholders by entities. A *ground instance of a rule* is a rule obtained by replacing all placeholders by entities. All occurrences of a placeholder must be replaced by the same entity. For example, the following is a ground instance of the rule mentioned above:

bornIn(*Einstein*, *Ulm*) ⇒ ¬ *bornIn*(*Einstein*, *Timbuktu*)

SOFIE's Rules. We have developed a number of rules for SOFIE. One of the rules states that a functional relation (e.g., the *bornIn* relation) should not have more than one argument for a given first argument:

R(*X*, *Y*)
∧ *type*(*R*, *function*)
∧ *different*(*Y*, *Z*)
⇒ ¬ *R*(*X*, *Z*)

The rule guarantees, for example, that people cannot be born in more than one place. Since *disambiguatedAs* is also a functional relation, the rule also guarantees that one wic is disambiguated to at most one entity. There are also other rules, some of which concern the textual facts. One rule asserts that if pattern P occurs with entities x and y and if there is a relation r such that $r(x, y)$, then P expresses r. For example, if the pattern "X was born in Y" appears with Albert Einstein and his true location of birth, Ulm, then it is likely that "X was born in Y" expresses the relation *bornInLocation*. A naive formulation of this rule looks as follows:

patternOcc(P, X, Y)
∧ *R*(*X*, *Y*)
⇒ *expresses*(*P*, *R*)

We need to take into account, however, that patterns hold between wics, whereas facts hold between entities. Our model allows incorporating this constraint in an elegant way:

patternOcc(P, WX, WY)
∧ *disambiguatedAs*(*WX*, *X*)
∧ *disambiguatedAs*(*WY*, *Y*)
∧ *R*(*X*, *Y*)
⇒ *expresses*(*P*, *R*)

There is a dual version of this rule: If the pattern expresses the relation r, and the pattern occurs with two entities x and y, and x and y are of the correct types, then $r(x, y)$:

patternOcc(P, WX, WY)
∧ *disambiguatedAs*(*WX*, *X*)
∧ *disambiguatedAs*(*WY*, *Y*)
∧ *domain*(*R*, *DOM*)
∧ *type*(*X*, *DOM*)
∧ *range*(*R*, *RAN*)
∧ *type*(*Y*, *RAN*)
∧ *expresses*(*P*, *R*)
⇒ *R*(*X*, *Y*)

By this rule, the pattern comes into play only if the two entities are of the correct type. Thus, the very same pattern can express different relations if it appears with different types. Another rule makes sure that the disambiguation prior influences the choice of disambiguation:

disambPrior(W, X, N)
⇒ *disambiguatedAs*(*W*, *X*)

Softness. The rules for SOFIE have to be designed manually. We believe that the rules we provided should be general enough to be useful with a large number of relations. More (relation-specific) rules can be added. In general, it is impossible to satisfy all of the these rules simultaneously. For example, as soon as there exist two disambiguation priors for the same wic, both will enforce a certain disambiguation. Two disambiguations, however, contradict the functional constraint of *disambiguatedAs*. This is why certain rules will have to be violated. Some rules are less important than others. For example, if a strong disambiguation prior requires a wic to be disambiguated as X, while a weaker prior desires Y, then X should be given preference – unless other constraints favor Y. This is why a sophisticated approach is needed to compute the most likely hypotheses.

3.3 MAX-SAT Model

SOFIE aims to find the hypotheses that should be accepted as true facts so that a maximal number of rules are satisfied. The problem can be cast into a *maximum satisfiability problem* (*MAX SAT problem*). In our setting, the variables are the hypotheses and the rules are transformed into propositional formulae on them. This view would allow violating some rules, but it would not allow weighting them. Therefore, we consider here a setting where the formulae are weighted. One approach for dealing with weighted first order logic formulae is Markov Logic [31]. Markov Logic, however, would lift the problem to a more complex level (that of inferring probabilities), usually involving heavy machinery. Furthermore, Markov Logic Networks might not be able to deal efficiently with the millions of facts that YAGO provides. Fortunately, there is a simpler option, which also allows dealing with weighted logic formulae: the weighted maximum satisfiability setting or Weighted MAX-SAT.

Weighted MAX-SAT. The weighted MAX-SAT problem is based on the notion of clauses:

DEFINITION 1: [*Clause*]
A *clause* C over a set of *variables* $\mathcal{X}$ consists of
(1) a *positive literal set* $c^1 = \{x_1^1, \ldots, x_n^1\} \subseteq \mathcal{X}$
(2) a *negative literal set* $c^0 = \{x_1^0, \ldots, x_m^0\} \subseteq \mathcal{X}$
A *weighted clause* over $\mathcal{X}$ is a clause C over $\mathcal{X}$ with an associated weight $w(C) \in \mathbb{R}^+$.

Given a clause C over a set $\mathcal{X}$ of variables, we say that a variable $x \in \mathcal{X}$ appears with *polarity* p in C, if $x \in C^p$. The semantics of clauses is given by the notion of assignments:

DEFINITION 2: [*Assignment*]
An *assigment* for a set $\mathcal{X}$ of variables is a function $v : \mathcal{X} \to \{0, 1\}$. A *partial assignment* for $\mathcal{X}$ is a partial function $v : \mathcal{X} \rightharpoonup \{0, 1\}$. A (partial) assignment for $\mathcal{X}$ *satisfies* a clause C over $\mathcal{X}$, if there is an $x \in \mathcal{X}$, such that $x \in C^{v(x)}$.

DEFINITION 3: [*Weighted MAX SAT*]
Given a set $\mathcal{C}$ of weighted clauses over a set $\mathcal{X}$ of variables, the *weighted MAX SAT problem* is the task of finding an assignment v for $\mathcal{X}$ that maximizes the sum of the weights of the satisfied clauses:

$$\sum_{c \in \mathcal{C} \text{ is satisfied in } v} w(c)$$

An assignment that maximizes the sum of the satisfied clauses in a weighted MAX SAT problem is called a *solution* of the problem.

Mapping SOFIE's Rules into Clauses. The task that SOFIE faces is, given a set of facts, a set of hypotheses and a set of rules, finding truth values for the hypotheses so that a maximum number of rules is satisfied. This problem can be cast into a weighted MAX-SAT problem as follows. We assume a finite set of rules, a finite set of ontological facts, and a finite set of textual facts. Together they implicitly define a finite set of entities. We map this model into clauses as follows:

1. Each rule is syntactically replaced by all of its grounded instances. Since the set of entities is finite, the set of ground instances is finite as well. (Section 4 will discuss efficient techniques for performing this lazily on demand, avoiding many explicit groundings.)

2. Each ground instance is transformed into one or multiple clauses as usual in propositional logic. The following rewriting template covers all rules introduced in Section 3.2: $p_1 \wedge \ldots \wedge p_n \Rightarrow c \quad \rightsquigarrow \quad (\neg p_1 \vee \ldots \vee \neg p_n \vee c)$

3. The set of all statements that appear in the clauses becomes the set of variables. Note that these statements will include not only the ontological facts and the textual facts, but also all hypotheses that the rules construct from them.

These steps leave us with a set of variables and a set of clauses.

Weighting. The clauses about the disambiguation of wics and the quality of patterns may possibly be violated. These are the clauses that contain the relation *patternOcc* or the relation *disambPrior* (see Section 3.2). We assign them a fixed weight w. For the *disambPrior* facts, we multiply w with the disambiguation prior, so that the prior analysis is reflected in the weight. The other clauses should not be violated. We assign them a fixed weight W. W is chosen so large that even repeated violation (say, hundred-fold) of a clause with weight w does not sum up to the violation of a clause with weight W. This way, every clause has a weight and we have transformed the problem into a weighted MAX-SAT problem.

Ockham's Razor. The optimal solution of the weighted MAX-SAT problem shall reflect the optimal assignment of truth values to the hypotheses. In practice, however, there are often multiple optimal solutions. In particular, some optimal solutions may make hypotheses true even if there is no evidence for them. For example, an optimal solution may assert that a pattern expresses a relation even if there are no examples for the pattern. This is because a rule of the form $A \Rightarrow B$ can always be satisfied by setting B to true, even if A (the evidence) is false. To avoid this phenomenon, we give preference to the solution that makes the least number of hypotheses true.[5] We encode this desideratum in our weighted MAX-SAT problem by adding a clause $(\neg h)$ with a small weight ε for each hypothesis h. This ensures that a hypothesis is made true only if there is evidence for it. Given that the number of hypotheses is huge, the desired solution will make only a very small portion of the hypotheses true. The exact value for ε is not essential. Given two solutions of otherwise equal weight, ε just serves to choose the one that makes the least number of hypotheses true.

[5]This principle is known as *Ockham's Razor*, after the 14th-century English logician William of Ockham. In our setting (as in reality), omitting this principle leads to random hypotheses being taken for true.

4. IMPLEMENTATION

SOFIE's main components are the pattern-extraction engine and the Weighted MAX-SAT solver. They are described in the next two subsections, followed by an explanation of how everything is put together into the overall SOFIE system.

4.1 Pattern Extraction

Pattern Occurrences. The pattern extraction component takes a document and produces all patterns that appear between any two entity names. First, the system *tokenizes* the document, splitting the document into short strings (*tokens*). The tokenization identifies and normalizes numbers, dates and, in Wikipedia articles, also Wikipedia hyperlinks. The tokenization employs lists (such as a list of stop words and a list of nationalities) to identify known words. The tokenization also identifies strings that must be person names.[6] The output of this procedure is a list of tokens. Next, "interesting" tokens are identified in the list of tokens. Since we are primarily concerned with information about individuals, all numbers, dates and proper names are considered "interesting". Whenever two interesting tokens appear within a window of a certain width, the system generates a *pattern occurrence fact*. More precisely, assume x and y are interesting words and appear in document d, separated by the sequence of tokens p. Then the following fact is produced:

$patternOcc(p,\ x@d,\ y@d)[1]$

Tokenizing Wikipedia. Wikipedia is a special type of corpus, because it also provides structured information such as infoboxes, lists, etc. Infoboxes and categories are tokenized as follows. The article entity (given by the title of the article) is inserted before each attribute name and before each category name. For example, the part "born in = Ulm" in the infobox about Albert Einstein is tokenized as "Albert Einstein born in = Ulm". By this minimal modification, these structured parts become largely accessible to SOFIE.

Disambiguation. Our system produces pattern occurrences with wics. Each wic can have several meanings. The system looks up the potential meanings in the ontology and produces a disambiguation prior for each of them. For example, suppose word w occurs in document d and w refers to the entities $e_1, \ldots, e_n$ in the ontology. Then, the system produces a fact of the following form for each e_i:

$disambPrior(w@d,\ e_i,\ l(d, w, e_i))[1]$

Here, $l(d, w, e_i)$ is a real value that expresses the likelihood that w means e_i in document d. There are numerous approaches for estimating this value [3]. We use a simple but effective estimation, known as the *bag of words approach*: Consider the set of words in d, and for each e_i, consider the set of entities connected to e_i in the ontology. We compute the intersection of these two sets and set $l(d, w, e_i)$ to the size of the intersection. This value increases with the amount of evidence that is present in d for the meaning e_i. We normalize all $l(d, w, e_i), i = 1 \ldots n$ to a sum of 1.

[6]The preprocessing tools are available at `http://mpii.de/~suchanek/downloads/javatools`.

We observe that this full disambiguation procedure is not always necessary. First, all literals in the document (such as dates) are already normalized. Hence, they are always unambiguous. Second, some words have only one meaning. For these tokens, our system produces no disambiguation prior. Instead, it produces pattern occurrences that contain the respective entity directly instead of the wic.

4.2 Weighted MAX-SAT Algorithm

Prior Assignments. In our weighted MAX-SAT problem, we have variables that correspond to hypotheses (such as *developed*(*Microsoft*, *JavaProgrammingLanguage*)) and variables that correspond to facts (namely ontological facts and textual facts). A solution to the weighted MAX-SAT problem should assign the truth value 1 to all previously existing facts. Therefore, we assign the value 1 to all textual facts and all ontological facts a priori. This assumes that the ontology is consistent with the rules. In the case of YAGO, used in all our experiments, this holds by the construction methods [38]. Furthermore, we assume that the ontology is complete on the *type* and *means* facts. For YAGO, this assumption is acceptable, because all entities in YAGO have *type* and *means* relations. If *type* and *means* are fixed, this allows certain simplifications, most importantly, precomputing the disambiguation prior (as explained in the previous subsection). This gives us a partial assignment, which already assigns truth values to a large number of statements.

Approximation Algorithms. The weighted MAX-SAT problem is NP-hard [20][7]. This means that it is impractical to find an optimal solution for large instances of the problem. Some special cases of the weighted MAX-SAT problem can be solved in polynomial time [21, 29]. However, none of them applies in our setting. Hence, we resort to using an approximation algorithm. An *approximation algorithm for the weighted MAX-SAT problem* produces an assignment for the variables that is not necessarily an optimal solution. The quality of that assignment is assessed by the *approximation ratio*, i.e., the weight of all clauses satisfied by the assignment divided by the weight of all clauses satisfied in the optimal assignment. An algorithm for the weighted MAX-SAT problem is said to have an *approximation guarantee* of $r \in [0,1]$, if its output has an approximation ratio greater than or equal to r for all weighted MAX-SAT problems. Many algorithms have only weak approximation guarantees, but perform much better in practice. Among the numerous approximation algorithms that appear in the literature (see [7]), we focus here on *greedy algorithms* for efficiency. Their run-time is linear or quadratic in the total size of the clauses.

Johnson's Algorithm. One of the most prominent greedy algorithms is *Johnson's Algorithm* [22]. It is particularly simple and has been shown to have an approximation guarantee of 2/3 [11]. However, the algorithm cannot produce assignments with an approximation ratio greater than 2/3 if the problem has the following shape [45]: For some integer k, the set of variables is $\mathcal{X} = \{x_1, \ldots, x_{3k}\}$ and the set of clauses is

$$\begin{aligned} &x_{3i+1} \vee x_{3i+2} \\ &x_{3i+1} \vee x_{3i+3} \\ &\neg x_{3i+1} \end{aligned} \qquad \text{for } i = 0, \ldots, k-1$$

[7]The SAT problem is not NP-hard if there are at most two literals per clause. The weighted and unweighted MAX-SAT problems, however, are NP-hard even when each clause has no more than two literals.

Unfortunately, this is exactly the shape of clauses induced by the rule for functional relations in the SOFIE setting (in negated form, see Section 3.2). The relation *disambiguatedAs* already falls into this category, making Johnson's Algorithm perform poorly for the very instances that are common in our case of interest. Therefore, we consider different greedy techniques that overcome this problem, and develop an algorithm that is particularly geared for the structure of clauses that typically occur in SOFIE.

FMS Algorithm. We introduce the *Functional Max Sat Algorithm* here, which is tailored for clauses of the above shape. The algorithm relies on *unit clauses*:

Definition 4: [*Unit Clauses*]
Given a set of variables $\mathcal{X}$, a partial assignment v on $\mathcal{X}$ and a set of clauses $\mathcal{C}$ on $\mathcal{X}$, a *unit clause* is a clause $c \in \mathcal{C}$ that is not satisfied in v and that contains exactly one unassigned literal.

Intuitively speaking, unit clauses are the clauses whose satisfaction in the current partial assignment depends only on a single variable. Our algorithm uses them as follows:

Algorithm 1: Functional Max Sat (FMS)
Input: Set of variables $\mathcal{X}$
Set of weighted clauses $\mathcal{C}$
Output: Assignment v for $\mathcal{X}$
1 $v :=$ the empty assignment
2 `WHILE` there exist unassigned variables
3 `FOR EACH` unassigned $x \in \mathcal{X}$
4 $m^0(x) := \sum \{ w(c) \mid c \in \mathcal{C}$ unit clause, $x \in c^0 \}$
5 $m^1(x) := \sum \{ w(c) \mid c \in \mathcal{C}$ unit clause, $x \in c^1 \}$
6 $x^* :=$ $\arg\max(|m^1(x) - m^0(x)|)$
breaking ties arbitrarily
7 $v(x^*) = m^1(x^*) > m^0(x^*)$? 1 : 0

Note that if there are no unit clauses, the algorithm will set an arbitrary unassigned variable to 0. If $m^0(x)$ and $m^1(x)$ are only recomputed for variables affected by the previous assignment, the FMS Algorithm can be implemented [35] to run in time $O(n \cdot m \cdot k \cdot \log(n))$, where n is the total number of variables in the clauses, k is the maximum number of variables per clause and m is the maximum number of appearances of a variable.

DUC Propagation. Once the algorithm has assigned a truth value to a single variable, the truth value of other variables may be implied by necessity. These variables are called *safe*:

Definition 5: [*Safe Variable*]
Given a set of variables $\mathcal{X}$, a partial assignment v on $\mathcal{X}$ and a weighted set of clauses $\mathcal{C}$ on $\mathcal{X}$, an unassigned variable $x \in \mathcal{X}$ is called *safe*, if

$$\sum_{\substack{c \text{ unit clause} \\ x \in c^p}} w(c) \quad \geq \quad \sum_{\substack{c \text{ unsatisfied clause} \\ x \in c^{\neg p}}} w(c)$$

for some $p \in \{0,1\}$. p is called the *safe truth value* of x.

It can be shown [35, 25, 44] that safe variables can be assigned their safe truth value without changing the weight of the best solution that can still be obtained. Iterating this procedure for all safe variables is called *Dominating*

Unit Clause Propagation (DUC Propagation). DUC Propagation subsumes the techniques of unit propagation and pure literal elimination employed by the DavisŰ-Putnam-ŰLogemannŰ-Loveland (DPLL) algorithm [12] for the SAT problem.

We enhance the FMS Algorithm by invoking DUC propagation in each iteration of the outer loop (i.e., before assigning truth values to previously unassigned variables). The resulting algorithm is coined FMS*. We prove [35]

THEOREM 1: [*Approximation Guarantee of FMS**]
The FMS* Algorithm has approximation guarantee 1/2.

Lazy Generation of Clauses. Our algorithm works on a database representation of the ontology. The hypotheses and the textual facts are stored in the database as well. Since our weighted MAX-SAT problem may be huge, we refrain from generating all clauses explicitly. Rather, we use a lazy technique that, given a statement s, generates all clauses in which s appears on the fly [35]. The algorithm returns only those clauses that are not yet satisfied. It uses an ordering strategy, computing ground instances of the most constraining literals first.

4.3 Putting Everything Together

SOFIE Algorithm. SOFIE operates on a given set of ontological facts (the ontology) and a given set of documents (the corpus). We run SOFIE in large batches, because hypothesis testing is more effective when SOFIE considers many patterns and hypotheses together.

SOFIE first parses the corpus, producing textual facts. These are mapped into clause form, together with the resulting hypotheses. Then, the FMS* Algroithm is run, assigning truth values to hypotheses. Afterwards, the true hypotheses can be accepted as new facts of the ontology. This applies primarily to new ontological facts (i.e., facts with relations such as *bornOnDate*). Going beyond the ontological facts, it is also possible to include the new *expresses* facts in the ontology. Thus, the ontology would store which pattern expresses which relation.

Now suppose that, later, SOFIE is run on a different corpus. Since the SOFIE algorithm assigns the truth value 1 to all facts from the ontology, the later run of SOFIE would adopt the *expresses* facts from the previous run. This way, SOFIE can already build on the known patterns when it analyzes a new corpus.

5. EXPERIMENTS

To study the accuracy and scalability of SOFIE under realistic conditions, we carried out experiments with different corpora, using YAGO [37] as the pre-existing ontology. YAGO contains about 2 million entities and 20 million facts for 100 different relations. Our experiments here demonstrate that SOFIE is able to enhance YAGO by adding previously unharvested facts and completely new facts without degrading YAGO's high accuracy.

We experimented with both semi-structured sources from Wikipedia and with unstructured free-text sources from the Web. In each of these two cases, we first perform controlled experiments with a small corpus for which we could manually establish the ground truth of all correct and potentially extractable facts. We report both precision and recall for these controlled experiments. Then, for each of the semi-structured and unstructured cases, we show results for large-scale variants, with evaluation of output precision and run-time. Recall is not the primary focus of SOFIE, especially since we may hope for redundancy in large corpora. All experiments were performed with the rules from Section 3.2, unless otherwise noted. In all cases, we used the default weights $W = 100$ for the inviolable rules, $w = 1$ for the violable rules and $\varepsilon = 0.1$ for Ockham's Razor. The experiments were run on a standard desktop machine with a 3 GHz CPU and 3.5 GB RAM, using the PostgreSQL database system.

5.1 Semi-Structured Sources

5.1.1 Controlled Experiment

To study the performance of SOFIE on semi-structured text under controlled conditions, we created a corpus of 100 random Wikipedia articles about newspapers. We decided for 3 relations that were not present in YAGO, and added 10 instances for each of them as seed pairs to YAGO. SOFIE took 3 min to parse the corpus, and 5 min to compute the truth values for the hypotheses. We evaluated SOFIE's precision and recall manually, as shown in Table 1.

Table 1: Results on the Newspaper Corpus (1)

Relation	Ground truth	Output pairs	Correct pairs	Precision	Recall
foundedOnDate	89	87	87	100%	97.75%
hasLanguage	45	29	28	96.55%	62.22%
ownedBy	57	49	49	100%	85.96%

Thanks to its powerful tokenizer, SOFIE immediately finds the infobox attributes for the relations (such as *owner* for the owner of a newspaper). In addition, SOFIE finds some facts from the article text, but not all (e.g., for *hasLanguage*). As our results show, SOFIE can achieve a precision that is similar to the precision of tailored and specifically tuned infobox harvesting methods as employed in [37, 38, 4, 42]. To test the performance of SOFIE without infoboxes, we removed the infoboxes from half of the documents, the goal being to extract the now missing attributes from other parts of the articles. We chose our seed pairs from the portion of articles that did have infoboxes. Table 2 shows the results.

Table 2: Results on the Newspaper Corpus (2)

Relation	Ground truth	Output pairs	Correct pairs	Precision	Recall
foundedOnDate	89	78	77	98.71%	86.51%
hasLanguage	45	18	18	100%	40.00%
ownedBy	57	26	26	100%	45.76%

Recall is much lower if the infoboxes are not present. Still, SOFIE manages to find information also in the articles without infoboxes. This is because SOFIE finds the category *"Newspapers established in..."*. This category indicates the year in which the newspaper was founded. Interestingly, this category did not occur in our seed pairs for *foundedOnDate*. Thus, SOFIE had no clue about the quality of this pattern. By help of the infoboxes, however, SOFIE could establish a large number of instances of *foundedOnDate*. Since many of these had the category *"Newspapers established in..."*, SOFIE accepted also the category pattern *"Newspapers established in X"* as a good pattern for the relation *foundedOnDate*. In other words, newly found instances of the target relation induced the acceptance of new patterns, which in turn produced new instances. This principle is very close to what has been proposed for DIPRE [10]

and Snowball [2]. However, in contrast to such prior work, SOFIE achieves this effect without any special consideration, simply by its principle of including patterns and hypotheses in its reasoning model. In the ideal case, SOFIE could extract the information solely from the article text, thus abandoning the dependence on infoboxes. Then, SOFIE would perform a task similar to KYLIN [42]. Up to now, however, the performance of SOFIE on this task trails behind KYLIN, which has a recall of over 90%. This is because KYLIN is highly tuned and specifically tailored to Wikipedia, whereas SOFIE is a general-purpose information extractor.

5.1.2 Large-Scale Experiment

We created a corpus of 2000 randomly selected Wikipedia articles. We chose 13 relations that are frequent in YAGO. We added a rule saying that the birth date and the death date of a person shall not have a difference of more than 100 years. For simplification, we also added a rule saying that a person cannot be both an actor and a director of a movie. This setting poses a stress test to SOFIE because of the high thematic diversity: Articles could be "out of scope" (relative to the 13 target relations) and even an individual article could cover very heterogeneous topics; these difficulties can mislead any IE method. SOFIE took 1:27 hours to parse the corpus. It took 12 hours to create all hypotheses, and the actual FMS* Algorithm ran for 1 hour and 17 min. Table 3 shows the results of our manual evaluation (where we always disregard facts that were already known to YAGO).

Table 3: Results on the Wikipedia Corpus

Relation	Output pairs	Correct pairs	Prec.
actedIn	8	8	100%
bornIn	122	116	95.08%
bornOnDate	119	115	96.63%
diedOnDate	20	19	95.00%
directed	8	10	80.00%
establishedOnDate	50	44	88.00%
hasArea	1	1	100%
hasDuration	1	1	100%
hasPopulation	20	18	90.00%
hasProductionLanguage	4	4	100%
hasWonPrize	35	34	97.14%
locatedIn	109	100	91.74%
writtenInYear	8	8	100%
Total	505	478	94.65%

The evaluation shows good results. However, the precision values are slightly worse than in the small-scale experiment. This is due to the thematic diversity in our corpus. The documents comprised articles about people, cities, movies, books and programming languages. Our relations, in contrast, mostly apply only to a single type each. For example, *bornOnDate* applies exclusively to people. Thus, the chances for examples and counterexamples for each single relation are lowered. Still, the precision values are very good. For the *bornIn* relation, SOFIE found the category pattern "People from X". In most cases, this category indeed identifies the birth place of people. In some cases, however, the category tells where people spent their childhood. This misleads SOFIE. Overall, the patterns stemmed from the article texts, the categories, and the infoboxes. So SOFIE harvested both the semi-structured and the unstructured part of Wikipedia in a unified manner. Given this general-purpose nature of SOFIE, the results are remarkably good.

5.2 Unstructured Web Sources

5.2.1 Controlled Experiment

To study the performance of SOFIE on unstructured text under controlled conditions, we used a corpus of newspaper articles that has been used for a prototypical IE system, Snowball [2]. Snowball was run on a collection of some thousand documents. For a small portion of that corpus, the authors established the ground truth manually. For copyright reasons, we only had access to this small portion. It comprises 150 newspaper articles. The author kindly provided us with the output of Snowball on this corpus. The corpus targets the *headquarters* relation, which is of particular finesse, as city names are usually highly ambiguous. To exclude the effect of the ontology in SOFIE, we manually added all organizations and cities mentioned in the articles to YAGO. This gives us a clean starting condition for our experiment, in which all failures are attributed solely to SOFIE and not to the ontology. As the *headquarters* relation is not known to YAGO, we added 5 pairs of an organization and a city as seed pairs to the ontology. Unlike Snowball, SOFIE extracts *disambiguated* entities. Hence, we disambiguated each name in the ground truth manually. We expect SOFIE to disambiguate its output correctly, whereas we will count any surface representation of the ground truth entity as correct for Snowball.

To run SOFIE with minimal background knowledge, we first ran it only with the *isHeadquartersOf* relation. This relation is not a function, so that SOFIE has no counterexamples. SOFIE took 2 minutes to parse the corpus, 22 minutes to create the hypotheses and 20 sec for the FMS* algorithm. We evaluated by the ideal metrics [2], which only takes into account "relevant pairs", i.e., pairs that have as a first component a company that appears in the ground truth. Table 4 shows results for Snowball and SOFIE (as SOFIE 1).

Table 4: Results on the Snowball Corpus

	Ground truth	Output pairs	Relev pairs	Correct pairs	Precision (ideal)	Recall
Snowball	120	429	65	37	56.69%	30.89%
SOFIE 1	120	35	35	32	91.43%	24.32%
SOFIE 2	120	46	46	42	91.30%	31.08%

SOFIE achieves a much higher precision than Snowball – even though SOFIE faced the additional task of disambiguation. In fact, the 3 cases where SOFIE fails are difficult cases of disambiguation, where *"Dublin"* does not refer to the Irish capital, but to a city in Ohio. To see how semantic information influences SOFIE, we added the original *headquarteredIn* relation, which is the inverse relation of *isHeadquartersOf*. We added a rule stating that whenever X is the headquarters of Y, Y is headquartered in X. Furthermore, we made *headquarteredIn* a functional relation, so that one organization is only headquartered in one location. The results are shown as SOFIE 2 in Table 4. The inverse relation has allowed SOFIE to find patterns, in which the organization precedes the headquarter ("Microsoft, a Redmond-based company"). This has increased SOFIE's recall to the level of Snowball's recall. At the same time, the functional constraint has kept SOFIE's precision at a very high level.

5.2.2 Large Scale Experiment

To evaluate SOFIE's performance on a large, unstructured corpus, we downloaded 10 biographies for each of 400 US senators, as returned by a Google search (less, if the pages

could not be accessed or were not in HTML). We excluded pages from Wikipedia. This resulted in 3440 HTML files. Extracting information from these files is a particulary challenging endeavor, because the documents are arbitrary, unstructured documents from the Web, containing, for example, tables, lists, advertisements, and occasionally also error messages. The disambiguation is particularly difficult. For example, there was one senator called James Watson, but YAGO knows 13 people with this name.

We added a rule saying that the birth date and the death date of a person shall not have a difference of more than 100 years. As explained in Section 4.3, we ran SOFIE in 5 batches of 20,000 pattern occurrences, keeping the true hypotheses and the patterns from the previous iteration for the next one. Overall, SOFIE took 7 hours to parse the corpus and 9 hours to compute the true hypotheses. We evaluated the results manually by checking each fact on Wikipedia, thereby also checking whether the entities have been disambiguated correctly. Table 5 shows our results.

Table 5: Results on the Biography Corpus

Relation	# Output pairs	# Correct pairs	Precision
politicianOf	339	≈ 322	94.99%
bornOnDate	191	168	87.96%
bornIn	119	104	87.40%
diedOnDate	66	65	98.48%
diedIn	29	4	13.79%
Total	744	673	90.45%

For *politicianOf*, we evaluated only 200 facts, extrapolating the number of correct pairs and the precision accordingly. Our evaluation shows very good results. SOFIE did not only extract birth dates, but also birth places, death dates, and the states in which the people worked as politicians. Each of these facts comes with its particular disambiguation problems. The place of birth, for example, is often ambiguous, as many cities in the United States bear the same name. Even the birth date may come with its particular difficulties if the name of the person refers to multiple people. Thus, we can be extremely satisfied with our precision values.

SOFIE could not establish the *diedIn* facts correctly, though. This is due to some misleading patterns that got established in the first batch. Counterexamples were only found in a later batch, when the patterns were already accepted. However, the general accuracy of SOFIE is still remarkable, given that the system extracted disambiguated, clean canonicalized facts from Web documents.

5.3 Comparison of MAX-SAT Algorithms

To see how the FMS* Algorithm performs in our SOFIE setting, we ran the algorithm on a small corpus of 250 biography files. We compared the FMS* Algorithm to Johnson's Algorithm [22] and to a simple greedy algorithm that sets a variable to 1 if the weight of unsatisfied clauses in which the variable occurs positive is larger than the weight of unsatisfied clauses where it appears negative. Table 6 shows the results. The number of unsatisfied inviolable clauses was 0 in all cases. In general, all algorithms perform very well. However, the FMS* Algorithm manages to satisfy the largest number of rules. It violates only one tenth of the rules that the other algorithms violate.

Table 6: MAX SAT Algorithms (SOFIE Setting)

Algorithm	Time	Unsatisfied violable clauses (of 172,165)	Weight of unsatisfied clauses (% of total)
FMS*	15 min	241	0.0013
Johnson	7 min	2,357	0,0301
Simple	7 min	2,583	0.0365

To study the performance of the FMS* Algorithm on general MAX-SAT problems, we used the benchmarks provided by the International Conference on Theory and Applications of Satisfiability Testing[8]. We took all benchmark suites where the optimal solution was available: (1) randomly generated weighted MAX-SAT problems with 2 variables per clause (90 problems), (2) randomly generated weighted MAX-SAT problems with 3 variables per clause (80 problems) and (3) designed weighted MAX-SAT problems (geared for "difficult" optimum solutions) with 3 variables per clause (15 problems). Each problem has around 100 variables and around 600 clauses. Table 7 shows the results.

Table 7: MAX SAT Algorithms (Benchmarks)

Algorithm	Averaged approximation ratios, %		
	Suite 1	Suite 2	Suite 3
Johnson	86.6837	91.5369	99.9682
Simple	86.6919	91.4946	99.9682
FMS*	87.3069	92.2848	99.9702

All algorithms find good approximate solutions, with approximation ratios on average greater than 85%. The setting of benchmarks is somewhat artificial and not designed for approximate algorithms. However, the experiments give us confidence that the FMS* Algorithm has at least comparable performance to Johnson's Algorithm. Our main goal, however, was to devise an algorithm that performs well in the SOFIE setting. The approximation guarantee of 1/2 gives a lower bound on the performance in the general case.

6. CONCLUSION

The central thesis of this paper is that the knowledge of an existing ontology can be harnessed for gathering and reasoning about new fact hypotheses, thus enabling the ontology's own growth. To prove this point, we have presented the SOFIE system that reconciles pattern-based information extraction, entity disambiguation, and ontological consistency constraints into a unified framework. Our experiments with both Wikipedia and natural-language Web sources have demonstrated that SOFIE can achieve its goals of harvesting ontological facts with very high precision.

For our experiments, we have used the YAGO ontology. However, we are confident that SOFIE could work with any other ontology that can be expressed in first order logic. SOFIE's main algorithm is completely source-independent. There is no feature engineering, no learning with cross validation, no parameter estimation, and no tuning of algorithms. Notwithstanding this self-organizing nature, SOFIE's performance could be further boosted by customizing its rules to specific types of input corpora. With appropriate rules, SOFIE could potentially even accommodate other IE paradigms within its unified framework, such as co-occurrence analysis [13] or infobox completion [42].

[8] `http://www.maxsat07.udl.es/`

7. REFERENCES

[1] Approximating the value of two power proof systems, with applications to max 2sat and max dicut. In *ISTCS* 1995.

[2] E. Agichtein, L. Gravano. *Snowball*: Extracting relations from large plain-text collections. In *ICDL* 2000.

[3] E. Agirre, P. Edmonds. *Word Sense Disambiguation: Algorithms and Applications (Text, Speech and Language Technology)*. Springer, 2006.

[4] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, Z. G. Ives. Dbpedia: A nucleus for a Web of open data. In *ISWC* 2007.

[5] M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, O. Etzioni. Open information extraction from the Web. In *IJCAI* 2007.

[6] M. Banko, O. Etzioni. Strategies for lifelong knowledge extraction from the web. In *K-CAP* 2007.

[7] M. Battiti, R. Protasi. Approximate algorithms and solutions for Max SAT. In G. Xue, editor, *Handbook of Combinatorial Optimization* , Kluwer, 2001.

[8] S. Blohm and P. Cimiano. Using the Web to reduce data sparseness in pattern-based information extraction. In *PKDD* 2007.

[9] S. Blohm, P. Cimiano, E. Stemle. Harvesting relations from the Web-quantifiying the impact of filtering functions. In *AAAI* 2007.

[10] S. Brin. Extracting patterns and relations from the World Wide Web. In *Selected papers from the Int. Workshop on the WWW and Databases*, 1999.

[11] J. Chen, D. K. Friesen, H. Zheng. Tight bound on Johnson's algorithm for maximum satisfiability. *J. Comput. Syst. Sci.*, 58(3):622–640, 1999.

[12] M. Davis, G. Logemann, D. Loveland. A machine program for theorem-proving. *Commun. ACM*, 5(7):394–397, 1962.

[13] V. de Boer, M. van Someren, B. J. Wielinga. Extracting instances of relations from Web documents using redundancy. In *ESWC* 2006.

[14] G. de Melo, F. M. Suchanek, A. Pease. Integrating YAGO into the Suggested Upper Merged Ontology. In *ICTAI* 2008.

[15] P. DeRose, W. Shen, F. Chen , A. Doan, R. Ramakrishnan. Building structured Web community portals: A top-down, compositional, and incremental approach. In *VLDB* 2007.

[16] O. Etzioni, M. Banko, M. J. Cafarella. Machine reading. In *AAAI* 2006.

[17] O. Etzioni, M. J. Cafarella, D. Downey, S. Kok, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, A. Yates. Web-scale information extraction in KnowItAll. In *WWW* 2004.

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

[19] W. A. Gale, K. W. Church, D. Yarowsky. One sense per discourse. In *HLT* 1991.

[20] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1979.

[21] B. Jaumard and B. Simeone. On the complexity of the maximum satisfiability problem for horn formulas. *Inf. Process. Lett.*, 26(1):1–4, 1987.

[22] D. S. Johnson. Approximation algorithms for combinatorial problems. *J. Comput. Syst. Sci.*, 9(3):256–278, 1974.

[23] D. Lenat, R.V. Guha. Building Large Knowledge Based Systems: Representation and Inference in the Cyc Project. Addison-Wesley, 1989.

[24] C. D. Manning and H. Schutze. *Foundations of Statistical NLP*. MIT Press, 1999.

[25] R. Niedermeier and P. Rossmanith. New upper bounds for maximum satisfiability. *Journal of Algorithms*, 36:2000, 2000.

[26] I. Niles and A. Pease. Towards a standard upper ontology. In *FOIS*, 2001.

[27] S. P. Ponzetto and M. Strube. Deriving a large-scale taxonomy from Wikipedia. In *AAAI*, 2007.

[28] H. Poon and P. Domingos. Joint inference in information extraction. In *AAAI*, 2007.

[29] V. Raman, B. Ravikumar, S. S. Rao. A simplified NP-complete MAXSAT problem. *Inf. Process. Lett.*, 65(1):1–6, 1998.

[30] F. Reiss, S. Raghavan, R. Krishnamurthy, H. Zhu, S. Vaithyanathan. An algebraic approach to rule-based information extraction. In *ICDE* 2008.

[31] M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62(1-2), 2006.

[32] S. Sarawagi. Information Extraction. *Foundations and Trends in Databases*, 2(1), 2008.

[33] W. Shen, A. Doan, J. F. Naughton, R. Ramakrishnan. Declarative information extraction using datalog with embedded extraction predicates. In *VLDB* 2007.

[34] S. Staab and R. Studer, editors. *Handbook on Ontologies, 2nd edition*. Springer, 2008.

[35] F. M. Suchanek. Automated Construction and Growth of a Large Ontology. *PhD thesis*, Saarland University, Germany, 2008.

[36] F. M. Suchanek, G. Ifrim, G. Weikum. Combining linguistic and statistical analysis to extract relations from Webdocuments. In *KDD*, 2006.

[37] F. M. Suchanek, G. Kasneci, G. Weikum. YAGO: A Core of Semantic Knowledge. In *WWW* 2007.

[38] F. M. Suchanek, G. Kasneci, G. Weikum. YAGO: A Large Ontology from Wikipedia and WordNet. *Elsevier Journal of WebSemantics*, 2008.

[39] L. Trevisan, G. B. Sorkin, M. Sudan, D. P. Williamson. Gadgets, approximation, linear programming. *SIAM J. Comput.*, 29(6):2074–2097, 2000.

[40] O. Udrea, L. Getoor, R. J. Miller. Leveraging data and structure in ontology integration. In *SIGMOD* 2007.

[41] G. Wang, Y. Yu, H. Zhu. Pore: Positive-only relation extraction from Wikipedia text. In *ISWC*, 2007.

[42] F. Wu and D. S. Weld. Autonomously semantifying Wikipedia. In *CIKM* 2007.

[43] F. Wu and D. S. Weld. Automatically refining the Wikipedia infobox ontology. In *WWW* 2008.

[44] Z. Xing and W. Zhang. MaxSolver: an efficient exact algorithm for (weighted) maximum satisfiability. *Artificial Intelligence*, 164(1-2):47–80, 2005.

[45] M. Yannakakis. On the approximation of maximum satisfiability. In *SODA* 1992.