

SOFIE: A Self-Organizing
Framework for Information
Extraction

Fabian M. Suchanek, Mauro Sozio,
Gerhard Weikum

MPI-I-2008-5-004 November 2008

Authors' Addresses

Fabian M. Suchanek
Max-Planck-Institute for Computer Science
Campus E1 4
66123 Saarbrücken
Germany

Mauro Sozio
Max-Planck-Institute for Computer Science
Campus E1 4
66123 Saarbrücken
Germany

Gerhard Weikum
Max-Planck-Institute for Computer Science
Campus E1 4
66123 Saarbrücken
Germany

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 near-perfect output, even from unstructured Internet documents.

Keywords

Ontologies, Information Extraction, YAGO, Reasoning

Contents

1	Introduction	3
1.1	Background and Motivation	3
1.2	Example Scenario	4
1.3	Contribution	4
2	Related Work	6
3	Model	9
3.1	Statements	9
3.2	Rules	12
3.3	MAX SAT Model	15
4	Implementation	20
4.1	Pattern Extraction	20
4.2	Weighted MAX SAT Algorithm	21
4.3	Putting Everything Together	27
5	Experiments	29
5.1	Semi-Structured Sources	29
5.1.1	Controlled Experiment	29
5.1.2	Large-Scale Experiment	31
5.2	Unstructured Web Sources	32
5.2.1	Controlled Experiment	32
5.2.2	Large-Scale Experiment	33
5.3	Comparison of MAX SAT Algorithms	34
6	Conclusion	36
A	Approximation Guarantee of FMS	37
B	Simple MAX SAT Algorithm	39

C Safe Variables	41
D Approximation Guarantee of FMS*	44

1 Introduction

1.1 Background and Motivation

Recently, several projects, such as YAGO [37, 38], Kylin/KOG [42, 43], and DBpedia [4], have successfully used Information Extraction (IE) methods for constructing large ontologies. They leveraged high-quality hand-crafted sources with latent knowledge, most notably, Wikipedia, for collecting individual entities and facts, and combined these results with a taxonomical hierarchy like WordNet [18], or SUMO [26, 14]. The focus has been on exploiting semi-structured components in Wikipedia like infoboxes and the category system, and free-text-based IE has been used only in an auxiliary role. The resulting ontologies contain millions of entities and tens of millions of facts (i.e., instances of relations between entities), and are organized in a consistent manner by a transitive and acyclic subclass relation. Moreover, empirical assessment has shown that these approaches have achieved an accuracy above 95 percent; so they are close to the best hand-crafted knowledge bases (which may also include false facts because of human mistakes).

Further expanding such automatically compiled ontologies and maintaining them as knowledge keeps evolving, would be a natural next stage but faces formidable challenges. Wikipedia’s semi-structured knowledge is huge but limited; for broader coverage, natural-language text sources, such as news articles, biographies, scientific publications, and also the full text of Wikipedia articles must be brought into scope. 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 text-analytic 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: use trusted facts as a basis for generating good text patterns that can guide the free-text IE, and scrutinize the resulting hypotheses 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. This, in turn, makes it deduce that Elvis went to school in Memphis.¹

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 based on linguistic patterns as positive evidence. The accuracy of pattern-based IE critically depends on having a variety of good patterns, and can be further boosted by also having counter-patterns for pruning false hypotheses. Finding strongly indicative positive and negative patterns is a key task of the IE process itself.

¹This is actually true. Albert Einstein went to secondary school in Switzerland, not Germany.

Entity disambiguation: Text-based IE recognizes words or phrases as indicators of named entities, but faces ambiguous choices in many cases, e.g., “Paris” denoting either the French capital or Paris in Texas. With many location names, companies, or product names, this is a difficult task by itself. As part of the IE process, mistakes in this step may mislead and degrade the entire process.

Consistency checking: Testing the plausibility of new hypotheses by checking their consistency with an existing ontology is a potentially powerful means for eliminating false results. This issue has been studied for integrating several ontologies (e.g., [40]), but this paper’s setting is very different by comparing a large set of IE-provided noisy candidates against a trusted core of facts. A conservative solution would be straightforward; however, the difficulty lies in achieving decent recall and identifying the best “survivors” from the candidate pool.

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, consistency checking, and eventually the identification of the best hypotheses for new facts;
- 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 finally, Section 5 discusses experiments.

2 Related Work

Fact Gathering. Unlike manual approaches such as WordNet [18], Cyc [23] or SUMO [26], *Information extraction (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 merely extract *non-canonical facts*. This means (1) that they do not disambiguate words to entities and (2) that they may extract no 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]. 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: infoboxes and the category system. Our new approach in the current paper, on the other hand, can process the full body of Wikipedia articles and is not tied at all to Wikipedia but can handle arbitrary Web pages and natural-language texts.

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. PORE does not incorporate world knowledge, which would be necessary for ontology building and extension.

Probably the most advanced solutions along these lines are currently provided by Kylin and KOG. Whenever an infobox type includes an attribute in some articles but this attribute has no value for a given article, Kylin analyzes the full text of the article to derive the most likely value. 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 can even discover new relation types. It builds on the class system of YAGO and DBpedia (along with the entities in each class) to train its learning algorithms for generating the subsumption graph between classes. Both Kylin and KOG are customized and optimized for Wikipedia articles; fact gathering from arbitrary Web sources has not been an objective.

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 information extraction methods of the system, however, have to be designed manually. The approach has been successfully applied for building and maintaining community portals like DBlife [15]; universal ontologies are not in the scope of this 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.

A fundamental approach that combines probabilistic graphical models with first-order logic is Markov Logic Networks [31]. Poon et al. [28] use them for simultaneously segmenting bibliographic entries and reconciling the involved entities (authors). 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 probabil-

ity 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], most notably, using – different – variations of SAT solvers, but follows a very different architectural paradigm. SOFIE aims to identify the best subset of hypotheses that is consistent with the existing ontology and its constraints, rather than performing general-purpose probabilistic inferences on them. Moreover, SOFIE is particularly geared for extracting instances of binary relations, rather than segmenting token sequences.

Ontology Integration and Extension. The goal of the current paper is to provide means for automatically extending an ontology with 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 all instances of a certain relation. While these lists 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. The approach is based on statistical methods only and has not been tried out with individual entities. 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

SOFIE is designed to extend an existing ontology. Hence, in the following, we assume a given ontology. For our experiments, we used YAGO, but our approach is open to any ontology. SOFIE extracts new information from text documents. Hence, in the following, we assume a given corpus of documents. In our experiments, we used documents downloaded from the Internet, but any other source could be chosen. SOFIE casts the information extraction into a logical reasoning problem. We will first introduce the notion of *statements* and then proceed to the notion of *rules*. Finally, we will show how the model can be cast into a Weighted MAX SAT problem.

3.1 Statements

Wics. As a knowledge gathering system, SOFIE has to address the problem of polysemy. In general, most words have several meanings. The word “*Java*”, for example, can refer to the programming language or to the Indonesian island. In a given context, however, a word is very likely to have only one meaning [19]. For example, the occurrences of the word “*Java*” in one Web document are likely to refer either all to the island or all to the programming language (unless the document is about the problems of polysemy). This gives rise to the following definition:

DEFINITION 1: [*Word in Context*]

A *word in context* (wic) is a pair of a word and a context.

For us, the context of a word will simply be the document in which the word appears. Thus, a wic is essentially a pair of a word and a document identifier¹. We use the notation *word@doc*. For example, we identify the word “*Java*” in the document *D8* by

¹A wic is related to a *KWIC* (keyword in context), also known as a *concordance*. A concordance is a word together with an ordered vector of the immediate surrounding

Java@D8

This way, the word “*Java*” in the document about Indonesian islands forms one wic, whereas the word “*Java*” in another document (be it about islands or programming languages) forms another wic. Once one occurrence of a wic has been disambiguated, it is assumed that all other occurrences of the wic are disambiguated as well. For example, once it has been figured out that document *D8* is about programming languages, all occurrences of *Java@D8* will be assumed to refer to the programming language. When talking about patterns in text documents, we will henceforth be precise and say that a pattern appears with two *wics* instead of two *words*. Following the all-embracing definition of entities, wics are also entities.

Facts and Hypotheses. For SOFIE, we will deal with relations of arbitrary arity. Hence we use a prefix notation for statements. Each statement can have an associated *truth value* of 1 or 0. We denote the truth value of a statement in square 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*. We will now see how both the ontology and the corpus can be interpreted as sources of facts.

Ontological Facts. SOFIE is designed to extend an existing ontology. We consider the ontology a set of facts. In the case of YAGO, this looks as follows: Technically speaking, the YAGO ontology is a reification graph. With our definitions, however, YAGO can also be interpreted as a set of facts. Here is an excerpt of YAGO in this light:

bornIn(AlbertEinstein, Ulm)[1]
bornOnDate(AlbertEinstein, 1879-03-14)[1]

...

This representation can also be enhanced by including the fact identifier of each fact as an additional argument. This would allow representing the *n*-ary relations of YAGO. For simplicity, however, we limit ourselves to the binary facts in YAGO for the time being.

words [24]. Thus, two occurrences of the same word in one document may form different concordances, but only one wic.

Textual Facts. SOFIE will extract textual information from the corpus. This information also takes the form of facts. One type of facts makes assertions about the number of times that a pattern occurred with two wics. For example, we might find that the pattern “X went to school in Y” occurred with the wics *Einstein@D29* and *Germany@D29*:

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

Another type of facts can state 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*. We will discuss later how the disambiguation prior can be computed. 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]

Other types of textual facts can be imagined. For example, the system could produce facts that tell which wic occurred in which document or which wic occurred how often with which other wic.

Hypotheses. Based on the ontological facts and the textual facts, SOFIE will form 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 will also hypothesize about whether a certain pattern expresses a certain relation:

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

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

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

Unifying Framework. By casting both the ontology and the corpus analysis into statements, SOFIE unifies the domains of ontology and information extraction. For SOFIE, there exist only statements. SOFIE will try 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 background knowledge to figure out which hypotheses are likely to be true. This knowledge takes the form of *rules*. Rules are based on *literals*:

DEFINITION 2: [*Literal*]

A *literal* is a statement that can have placeholders for the relation or some of the entities.

Here is an example of a literal with uppercase strings as placeholders:

$bornIn(X, Ulm)$

Now, a rule is basically a propositional formula over literals:

DEFINITION 3: [*Rule*]

A *rule* over a set of literals \mathcal{L} is one of the following

- an element of \mathcal{L}
- an expression of the form $\neg R$, where R is a rule over \mathcal{L}
- an expression of the form $(R_1 \diamond R_2)$, where R_1 and R_2 are rules over \mathcal{L} and $\diamond \in \{\wedge, \vee, \Rightarrow, \Leftrightarrow\}$.

As usual, we omit the outermost brackets and brackets around \wedge -expressions. With these conventions, the following line is a rule stating that whoever is born in Ulm is not born in Timbuktu:

$bornIn(X, Ulm) \Rightarrow \neg bornIn(X, Timbuktu)$

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. The relation between statements and literals is as follows:

DEFINITION 4: [*Ground Instances*]

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 one placeholder must be replaced by the same entity.

For example, the following is a ground instance of the rule mentioned above:

$$\text{bornIn}(\text{AlbertEinstein}, \text{Ulm}) \Rightarrow \neg \text{bornIn}(\text{AlbertEinstein}, \text{Timbuktu})$$

SOFIE’s Rules. We have developed a number of rules for SOFIE. One of the rules states that a functional relation should not have more than one second argument for a given first argument:

$$\begin{aligned} & R(X, Y) \\ \wedge & \text{type}(R, \text{function}) \\ \wedge & \text{different}(Y, Z) \\ \Rightarrow & \neg R(X, Z) \end{aligned}$$

The rule guarantees, for example, that people are not 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. In some sense, this rule is already employed during the construction of YAGO.

There are also other rules, some of which concern the textual facts. One rule says 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:

$$\begin{aligned} & \text{patternOcc}(P, X, Y) \\ \wedge & R(X, Y) \\ \Rightarrow & \text{expresses}(P, R) \end{aligned}$$

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

$$\begin{aligned} & \text{patternOcc}(P, WX, WY) \\ \wedge & \text{disambiguatedAs}(WX, X) \\ \wedge & \text{disambiguatedAs}(WY, Y) \\ \wedge & R(X, Y) \\ \Rightarrow & \text{expresses}(P, R) \end{aligned}$$

There exists also 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 $r(x, y)$:

$$\begin{aligned}
& \text{patternOcc}(P, WX, WY) \\
\wedge & \text{disambiguatedAs}(WX, X) \\
\wedge & \text{disambiguatedAs}(WY, Y) \\
\wedge & \text{domain}(R, DOM) \\
\wedge & \text{type}(X, DOM) \\
\wedge & \text{range}(R, RAN) \\
\wedge & \text{type}(Y, RAN) \\
\wedge & \text{expresses}(P, R) \\
\Rightarrow & R(X, Y)
\end{aligned}$$

By this rule, we are making a design choice: 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. We will see in the experiments (Section 5) how this works in practice. Another rule makes sure that the disambiguation prior influences the choice of disambiguation:

$$\begin{aligned}
& \text{disambPrior}(W, X, N) \\
\Rightarrow & \text{disambiguatedAs}(W, X)
\end{aligned}$$

Softness. 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. Before the next section discusses such a sophisticated approach, Figure 1 summarizes our notions again.

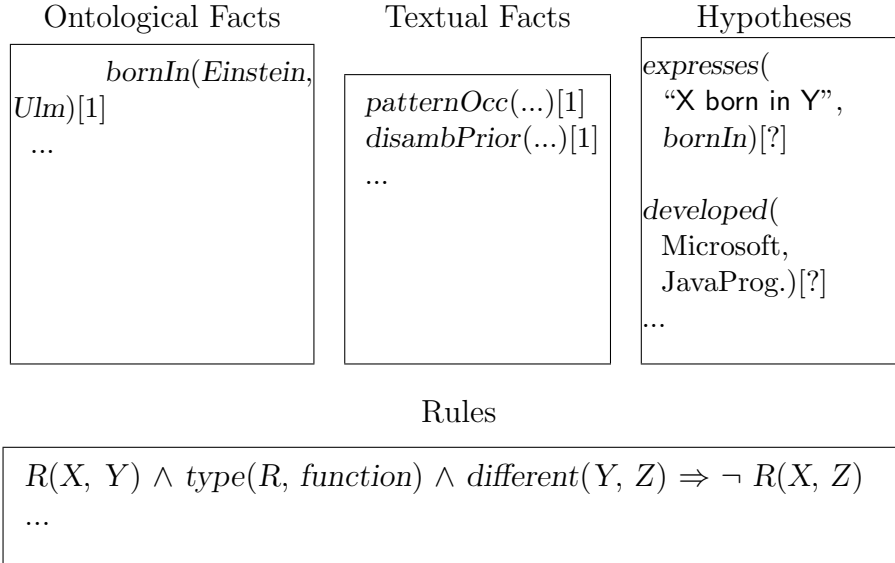


Figure 1: Statements and Rules

3.3 MAX SAT Model

Design Alternatives. Abstractly speaking, SOFIE aims to find the hypotheses that should be accepted as true so that a maximum number of rules is satisfied. Different approaches are conceivable. We sketch each of them informally before embarking on a formal definition of our approach.

- The problem can be cast into a rule-based setting, for example into a Datalog program (as it is done in [33]). However, this would not allow violating certain rules – which is a necessary desideratum.
- The problem can be cast into a *maximum satisfiability problem (MAX SAT problem)*. The MAX SAT problem is the task of, given a set of boolean variables and a set of propositional logic formulae, assigning truth values to the variables so that the number of satisfied formulae is maximized. In our setting, the variables would be the hypotheses and the rules would be transformed to propositional formulae on them. This view would allow violating some rules, but it would not allow weighting them.

- The problem can be cast into a Markov Logic Network [31]. A Markov Logic Network is concerned with a set of weighted first order logic formulae f_1, \dots, f_n over literals. For each formula f_i , the Markov Logic Network defines a function ϕ_i . ϕ_i takes as argument a possible *state of the world*, i.e., a possible assignment of truth values to the ground instances of the literals (e.g. $\{bornIn(Einstein, Ulm)=true, bornIn(Einstein, Timbuktu)=false, \dots\}$)². ϕ_i returns a real value that grows monotonically with the number of ground instances of f_i that are satisfied by the assignment. It can be shown that the product $\prod_i \phi_i$ defines a probability distribution over the possible states of the world. In particular, states that satisfy a higher number of formulae have a higher probability. Markov Logic Networks are very powerful and could be used to model our problem. However, such a model 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.

We chose a fourth option, which is simple, yet powerful enough to model our problem: the Weighted MAXimum satisfiability setting (Weighted MAX SAT).³

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

DEFINITION 5: [*Clause*]

A *clause* C over a set of *variables* \mathcal{X} consists of

- a *positive literal set* $c^1 = \{x_1^1, \dots, x_n^1\} \subseteq \mathcal{X}$
- a *negative literal set* $c^0 = \{x_1^0, \dots, x_m^0\} \subseteq \mathcal{X}$

We denote C by

$$(x_1^1 \vee x_2^1 \vee \dots \vee x_n^1 \vee \neg x_1^0 \vee \neg x_2^0 \vee \dots \vee \neg x_m^0)$$

A *weighted clause* over \mathcal{X} is a clause C over \mathcal{X} with an associated weight $w(C) \in \mathbb{R}^+$.

²See [31] for a more precise definition of literals, ground instances and formulae and for the necessary assumptions.

³In some sense, this problem is a special case of Markov Logic Networks, as computing the state of maximum likelihood in these networks is nothing else than solving a Weighted MAX SAT problem.

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 $x \in C^p$. Consider an example: If $\mathcal{X} = \{w, x, y, z\}$ is a set of variables, then the following is a clause over \mathcal{X} :

$$(w \vee x \vee \neg y \vee \neg z)$$

In this clause, w and x appear with positive polarity and y and z appear with negative polarity. Intuitively, the clause says that one of the variables w, x should be assigned a truth value of 1, while one of the variables y, z should be assigned a truth value of 0. This intuition is formalized as follows:

DEFINITION 6: [*Assignment, Partial Assignment, Satisfying Assignment*]
 An *assignment* for a set \mathcal{X} of variables is a function $v : \mathcal{X} \rightarrow \{0, 1\}$.
 A *partial assignment* for \mathcal{X} is a partial function $v : \mathcal{X} \rightarrow \{0, 1\}$. A (partial) assignment for \mathcal{X} *satisfies* a clause C over \mathcal{X} , if there is an $x \in \mathcal{X}$, such that $x \in C^{v(x)}$.

In the example, an assignment v with $v(w) = 1, v(x) = 1, v(y) = 0, v(z) = 1$ would be a satisfying assignment. We use the notation $\neg t = 1 - t$ for truth values t . Now we are ready to define the Weighted MAX SAT problem:

DEFINITION 7: [*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.

SOFIE. Intuitively speaking, the problem 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. Now, this problem can be cast into a Weighted MAX SAT problem. In the following, we assume a finite set of rules. Furthermore, we assume a finite set of ontological facts and a finite set of textual facts. These assumptions implicitly define a finite set of entities. We proceed as follows:

1. Every 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.

- Each ground instance is transformed to one or multiple clauses as usual in propositional logic. Here, we give a pattern that captures all rules introduced in Section 3.2:

$$p_1 \wedge \dots \wedge p_n \Rightarrow c \quad \rightsquigarrow \quad (\neg p_1 \vee \dots \vee \neg p_n \vee c)$$

- 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. Figure 2 exemplifies this process:


<i>Rule:</i>	$\text{bornIn}(X, \text{Ulm}) \Rightarrow \neg \text{bornIn}(X, \text{Timbuktu})$
<i>Ground instances:</i>	$\text{bornIn}(\text{Einstein}, \text{Ulm}) \Rightarrow \neg \text{bornIn}(\text{Einstein}, \text{Timbuktu})$ $\text{bornIn}(\text{Microsoft}, \text{Ulm}) \Rightarrow \neg \text{bornIn}(\text{Microsoft}, \text{Timbuktu})$...
<i>Clauses:</i>	$(\neg \text{bornIn}(\text{Einstein}, \text{Ulm}) \vee \neg \text{bornIn}(\text{Einstein}, \text{Timbuktu}))$ $(\neg \text{bornIn}(\text{Microsoft}, \text{Ulm}) \vee \neg \text{bornIn}(\text{Microsoft}, \text{Timbuktu}))$... 
<i>Variables:</i>	$\text{bornIn}(\text{Einstein}, \text{Ulm}), \text{bornIn}(\text{Einstein}, \text{Timbuktu}),$ $\text{bornIn}(\text{Microsoft}, \text{Ulm}), \text{bornIn}(\text{Microsoft}, \text{Timbuktu}),$...

Figure 2: Conversion to Weighted MAX SAT

Weighting. We partition the clauses into two sets as follows:

- 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 again 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.

2. 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. Among these, we prefer the solution that makes the least number of hypotheses true⁴. 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 makes sure that no hypothesis is made true if there is no evidence for it. The exact value for ε is not relevant. Given two solutions of otherwise equal weight, ε just serves to choose the one that makes the least number of hypotheses true.

⁴This 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 (strings that may denote) entities. First, the system *tokenizes* the document. The tokenization identifies (normalized) numbers, (normalized) dates and, in Wikipedia articles, also Wikipedia hyperlinks. Furthermore, the tokenization employs lists (such as a list of stop words and a list of nationalities) to identify known words. Last, the tokenization identifies strings that must be person names.¹ 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:

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

¹The preprocessing tools are available at <http://mpi.de/~suchanek/downloads/javatools>.

Tokenizing Wikipedia. Wikipedia is a special type of corpus, because it provides both unstructured text and structured parts like infoboxes, lists, etc. Infoboxes and lists of 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, \dots, e_n in the ontology. Then, the system produces a fact of the following form for each e_i :

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

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 $l(d, w, e_i), i = 1 \dots n$ to a sum of 1.

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 always refer to themselves. 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 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 case of YAGO, this is given by the construction methods. Furthermore, we will assume that the ontology is complete on the *type* and *means* facts. In case of 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, such as an a priori computation of the disambiguation prior (as explained in the previous section). This leaves us with a partial assignment, which already assigns truth values to a large number of statements.

Approximation. The Weighted MAX SAT problem is NP-hard [20]². This means that it is impractical to find an optimal solution for large instances of the problem, as it is the case in our setting. 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* is an algorithm that, given a Weighted MAX SAT problem, produces an assignment for its variables that is not necessarily an (optimal) solution. The quality of that assignment is assessed by the *approximation ratio*:

DEFINITION 8: [*Approximation Ratio of an Assignment*]

Given a Weighted MAX SAT problem in the form of a set \mathcal{X} of variables and a set \mathcal{C} of weighted clauses, and given a solution v_o , the *approximation ratio* of another assignment v for \mathcal{X} is the ratio

$$\sum_{\substack{c \in \mathcal{C} \\ c \text{ satisfied in } v}} w(c) \quad / \quad \sum_{\substack{c \in \mathcal{C} \\ c \text{ satisfied in } v_o}} w(c)$$

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 a weak approximation guarantee, but perform better in practice.

²The original 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.

Approximation Algorithms. The Weighted MAX SAT problem is an active area of research and numerous approximate algorithms have been proposed.³ In some special cases of the Weighted MAX SAT problem, the optimal solution can be approximated with a high approximation guarantee [39, 1]. However, our case is again too general. Out of the vast array of available methods, we focus on *greedy algorithms* here. This choice has two reasons: First, these methods are extremely simple and run in linear or quadratic time (in the total size of the clauses). Second, they will allow us to incorporate a resolution-like strategy in a straightforward way.

Greedy Algorithms. We call an algorithm *greedy*, if it assigns the variables incrementally without ever undoing its decision. 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 $\mathcal{X} = \{x_1, \dots, x_{3k}\}$ and the set of clauses is

$$\begin{array}{l} x_{3i+1} \vee x_{3i+2} \\ x_{3i+1} \vee x_{3i+3} \\ \neg x_{3i+1} \end{array} \quad \text{for } i = 0, \dots, k - 1$$

This, however, is exactly the shape of clauses induced by the rule for functional relations (in negation see Section 3.2). This means that Johnson’s Algorithm cannot solve SOFIE’s Weighted MAX SAT problem optimally, if instances of functional relations appear. Since already the relation *disambiguatedAs* falls into this category, Johnson’s Algorithm is less well suited for our problem. Hence, we consider a different greedy algorithm here.

FMS Algorithm. We introduce the *Functional Max Sat Algorithm* here, which is aimed at clauses induced by functional relations. The algorithm uses *unit clauses*:

DEFINITION 9: [*Unit Clause*]

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.

Strictly speaking, a unit clause is only defined with respect to an assignment. To simplify, we will occasionally not mention the assignment explicitly, when the assignment is clear from the context. Intuitively speaking, unit clauses

³See [7] for a survey.

are the clauses whose satisfaction in the current partial assignment depends only on one 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 **FOR EACH** $x \in \mathcal{X}$
- 3 $m^0(x) := \sum \{ w(c) \mid c \in \mathcal{C} \text{ unit clause, } x \in c^0 \}$
- 4 $m^1(x) := \sum \{ w(c) \mid c \in \mathcal{C} \text{ unit clause, } x \in c^1 \}$
- 5 $Q :=$ priority queue of all $x \in \mathcal{X}$,
ordered by descending $|m^1(x) - m^0(x)|$
- 6 **WHILE** Q is not empty
- 7 $x^* := Q.dequeue()$
- 8 $v(x^*) = [m^1(x^*) > m^0(x^*)]$
- 9 **FOR EACH** unassigned x s.t. $\exists C \in \mathcal{C} : x^* \in C, x \in C$
- 10 recompute $m^0(x), m^1(x)$, update priorities in Q

The algorithm takes as input a set of variables \mathcal{X} and a set of weighted clauses \mathcal{C} . In order to assign a truth value to a variable x , the algorithm considers only the unit clauses in which x appears. It computes for each variable x the sum of the weights of the unit clauses in which x appears with negative polarity ($m^0(x)$, line 3). Analogously, it computes the weights of the unit clauses in which x appears positive ($m^1(x)$, line 4). In case there are no unit clauses with x , both $m^0(x)$ and $m^1(x)$ are zero. In the loop (lines 6–10), the algorithm always picks the variable x that exhibits the largest difference of $m^0(x)$ and $m^1(x)$, breaking ties arbitrarily (line 7). In case there are no unit clauses at all, $m^0(x) = m^1(x) = 0$ for all $x \in Q$. In this case, an arbitrary variable x is dequeued from Q . If $m^1(x) > m^0(x)$, x is assigned the truth value 1. Else x is assigned the truth value 0 (line 8). After the assignment, the unit clauses in which x appeared are no longer unit clauses. However, new unit clauses might have sprung up. Hence, all variables affected by the assignment of x (line 9) have their values $m^0(x)$ and $m^1(x)$ recomputed. This changes their priority in the priority queue Q , so Q has to be updated (line 10). This procedure is iterated until all variables are assigned (lines 6–10). Assuming that all operations on the priority queue run in logarithmic time, the algorithm runs 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. We prove in

the Appendix A that the FMS Algorithm has an approximation guarantee of $1/2$:

THEOREM 1: [*Approximation Guarantee of the FMS Algorithm*]
 Independent of the order in which the variables are assigned, the FMS Algorithm has an approximation guarantee of $1/2$.

One might be tempted to construct a similar greedy algorithm that simply assigns the truth value t to a variable x , if the weight of unsatisfied clauses where x appears with polarity t exceeds the weight of unsatisfied clauses where x appears with polarity $\neg t$. We call this algorithm the *Simple Algorithm* and discuss it in Appendix B. In summary, it would have difficulties with the clauses in the SOFIE setting.

DUC Propagation. Once the FMS Algorithm has assigned a single variable, the truth value of others might be implied by necessity. These variables are called *safe*:

DEFINITION 10: [*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 there exists a truth value $t \in \{0, 1\}$ such that the weight of all unit clauses where x appears with polarity p is larger than the weight of all unsatisfied clauses in which x appears with polarity $\neg p$:

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

p is called the *safe truth value* of x .

The following theorem [25, 44] says that safe variables can be assigned their safe truth value without changing the weight of the best solution that can still be obtained:

THEOREM 2: [*Safe Truth Values are Safe*]

Let \mathcal{X} be a set of variables, let v be a partial assignment on \mathcal{X} , and let \mathcal{C} be a weighted set of clauses. Let x be a safe variable with safe truth value t . Let $v_o \supseteq v$ be an assignment that extends v and maximizes the sum of the weights of the satisfied clauses. Let v'_o be a variant of v_o that assigns t to x :

$$v'_o = v_o \setminus \{x \rightarrow 1, x \rightarrow 0\} \cup \{x \rightarrow t\}$$

Then

$$\sum_{c \in \mathcal{C} \text{ satisfied in } v'_o} w(c) \geq \sum_{c \in \mathcal{C} \text{ satisfied in } v_o} w(c)$$

We provide a proof in Appendix C. Theorem 2 gives rise to the technique of *Dominating Unit Clause Propagation*:

ALGORITHM 2: DUC Propagation

Input: Set of variables \mathcal{X}
Set of weighted clauses \mathcal{C}
Partial assignment v for \mathcal{X}

Output: Modified v

1 **WHILE** there exists a safe variable $x \in \mathcal{X}$
2 $v(x) :=$ safe truth value for x

Assigning one safe variable can create new unit clauses, which can give rise to new safe variables. Theorem 2 ensures that the safe variables can be assigned in any order without worsening the best solution that can still be achieved.⁴ In particular, the theorem ensures that if a partial assignment is part of an optimal solution, the enlarged assignment produced by the DUC Propagation will still be part of an optimal solution. Unlike the FMS Algorithm, however, DUC propagation does not necessarily produce a total assignment. Some variables may be left unassigned.

FMS Algorithm and DUC Propagation. We combine the FMS Algorithm with DUC propagation by calling the DUC propagation after each assignment (Algorithm 3).

⁴DUC Propagation subsumes the techniques of unit propagation and pure literal elimination employed by the Davis-Putnam-Logemann-Loveland (DPLL) algorithm [12] for the SAT problem.

ALGORITHM 3: FMS*

Input: Set of variables \mathcal{X}
Set of weighted clauses \mathcal{C}

Output: Assignment v for \mathcal{X}

- 1 Run Functional MAX SAT (Alg. 1) with \mathcal{X}, \mathcal{C}
- 2 After each assignment (line 8 in Functional MAX SAT):
- 3 Run DUC Propagation (Alg. 2) with $\mathcal{X}, \mathcal{C}, v$
- 4 Remove assigned variables from Q

We prove in Appendix D that this modification does not change the approximation guarantee of the algorithm:

THEOREM 3: [*Approximation Guarantee of the FMS* Algorithm*]

The FMS* Algorithm has an approximation guarantee of $1/2$.

The approximation guarantee does not improve over the approximation guarantee of the FMS Algorithm, because the presence of safe variables cannot be assumed in general.

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. 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. It operates in large batches, for efficiency and because the hypotheses testing is more effective when SOFIE considers many patterns and hypotheses together.

SOFIE first parses the corpus, producing textual facts. These are then compiled into hypotheses and mapped into clause form for the Weighted MAX SAT solver. Then, the FMS* Algorithm 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. All experiments were performed with the rules from Section 3.2, unless otherwise noted. In all cases, we used the weight $W = 100$ 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 three 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
<i>foundedOnD.</i>	89	87	87	100%	97.75%
<i>hasLanguage</i>	45	29	28	96.55%	62.22%
<i>ownedBy</i>	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
<i>foundedOnD.</i>	89	78	77	98.71%	86.51%
<i>hasLanguage</i>	45	18	18	100%	40.00%
<i>ownedBy</i>	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 the task performed by KYLIN [42]. Up to now, however, the performance of SOFIE on this task trails behind the performance of 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 posed 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.
<i>actedIn</i>	8	8	100%
<i>bornIn</i>	122	116	95.08%
<i>bornOnDate</i>	119	115	96.63%
<i>diedOnDate</i>	20	19	95.00%
<i>directed</i>	8	10	80.00%
<i>establishedOnDate</i>	50	44	88.00%
<i>hasArea</i>	1	1	100%
<i>hasDuration</i>	1	1	100%
<i>hasPopulation</i>	20	18	90.00%
<i>hasProductionLanguage</i>	4	4	100%
<i>hasWonPrize</i>	35	34	97.14%
<i>locatedIn</i>	109	100	91.74%
<i>writtenInYear</i>	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 to run the actual FMS*

algorithm. We evaluated according to 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

	Grnd. truth	Outp. pairs	Relev. pairs	Corr. pairs	Prec. (ideal)	Rec.
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. Adding the inverse relation has allowed SOFIE to find patterns, in which the organization precedes the headquarter (such as “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 particularly challenging endeavor, because the documents are arbitrary, unstructured pages 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 the results of the evaluation.

Table 5: Results on the Biography Corpus

Relation	# Output pairs	# Correct pairs	Precision
<i>politicianOf</i>	339	\approx 322	94.99%
<i>bornOnDate</i>	191	168	87.96%
<i>bornIn</i>	119	104	87.40%
<i>diedOnDate</i>	66	65	98.48%
<i>diedIn</i>	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 death places 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 Testing¹. 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.

¹<http://www.maxsat07.udl.es/>

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.

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].

Appendix A Approximation Guarantee of FMS

THEOREM 1: [*Approximation Guarantee of the FMS Algorithm*]
 Independent of the order in which the variables are assigned, the FMS Algorithm has an approximation guarantee of $1/2$.

Proof: We are given a Weighted MAX SAT problem in the form of a set \mathcal{X} of variables and a set \mathcal{C} of weighted clauses. The FMS algorithm constructs an assignment incrementally by assigning one variable after the other. We consider two sets, which we construct incrementally as the algorithm assigns the variables: The set $\mathcal{C}^- \subseteq \mathcal{C}$ will collect unsatisfied clauses, while the set $\mathcal{C}^+ \subseteq \mathcal{C}$ will collect satisfied clauses. In each step, the algorithm selects an unassigned variable $x \in \mathcal{X}$ and chooses the truth value t , if

$$\sum_{\substack{c \in \mathcal{C} \text{ unit clause} \\ x \in c^t}} w(c) \geq \sum_{\substack{c \in \mathcal{C} \text{ unit clause} \\ x \in c^{-t}}} w(c)$$

Before t is assigned to x , we update the sets \mathcal{C}^- and \mathcal{C}^+ as follows:

$$\begin{aligned} \mathcal{C}^+ &:= \mathcal{C}^+ \cup \{c \mid c \in \mathcal{C} \text{ unsatisfied clause, } x \in c^t\} \\ \mathcal{C}^- &:= \mathcal{C}^- \cup \{c \mid c \in \mathcal{C} \text{ unit clause, } x \in c^{-t}\} \end{aligned}$$

Every clause that is added to \mathcal{C}^+ will be satisfied by the current assignment of t to x . Every clause in \mathcal{C}^- will be unsatisfied and cannot be satisfied by future assignments. We observe that the clauses added to \mathcal{C}^+ have a higher total weight than the clauses added to \mathcal{C}^- . Hence, the update maintains the following invariance condition:

$$\sum_{c \in \mathcal{C}^+} w(c) \geq \sum_{c \in \mathcal{C}^-} w(c)$$

Each clause $c \in \mathcal{C}$ will be added to either one of the two sets during the course of the algorithm. Hence, when the algorithm terminates, \mathcal{C}^+ contains exactly

the satisfied clauses. Thus, the total weight of satisfied clauses achieved by the algorithm is:

$$\sum_{c \in \mathcal{C}^+} w(c)$$

Now consider an optimal solution v_o . Its weight is at most the weight of all clauses:

$$\sum_{c \in \mathcal{C}} w(c)$$

Hence the approximation ratio of the FMS Algorithm is

$$\begin{aligned} & \frac{\sum_{c \in \mathcal{C}^+} w(c)}{\sum_{c \in \mathcal{C}_{v_o}} w(c)} \\ \geq & \frac{\sum_{c \in \mathcal{C}^+} w(c)}{\sum_{c \in \mathcal{C}} w(c)} \\ \geq & \frac{\sum_{c \in \mathcal{C}^+} w(c)}{\sum_{c \in \mathcal{C}^-} w(c) + \sum_{c \in \mathcal{C}^+} w(c)} \\ \geq & \frac{\sum_{c \in \mathcal{C}^+} w(c)}{\sum_{c \in \mathcal{C}^+} w(c) + \sum_{c \in \mathcal{C}^+} w(c)} \\ \geq & \frac{1}{2} \end{aligned}$$

Appendix B Simple MAX SAT Algorithm

For completeness, we also examine a simple baseline algorithm for the Weighted MAX SAT problem here.

ALGORITHM 4: Simple Algorithm
Input: Set of variables \mathcal{X}
Set of weighted clauses \mathcal{C}
Output: Assignment v for \mathcal{X}
1 $v :=$ the empty assignment
2 **FOR EACH** $x \in \mathcal{X}$
3 $m^0 := \sum \{ w(c) \mid c \in \mathcal{C} \text{ unsatisfied, } x \in c^0 \}$
4 $m^1 := \sum \{ w(c) \mid c \in \mathcal{C} \text{ unsatisfied, } x \in c^1 \}$
5 $v(x) = [m^1 > m^0]$

This algorithm simply assigns the truth value t to a variable x , if the weight of unsatisfied clauses where x appears with polarity t exceeds the weight of unsatisfied clauses where x appears with polarity $\neg t$. By a similar argument as given in Appendix A, the Simple Algorithm also has an approximation guarantee of $1/2$. However, it can miss the optimal solution for the following type of Weighted MAX SAT problem: The set of variables is $\mathcal{X} = \{X, Y, Z\}$ and the clauses are

$$\begin{array}{ll} \neg X \vee \neg Y \vee Z & w_1 = W \\ X & w_2 = W - \varepsilon \\ Y \vee \neg Z & w_3 = W \end{array}$$

This constellation of clauses is quite frequent in the SOFIE setting, because most rules induce clauses of the shape $\neg X \vee \neg Y \vee Z$ (see Section 3.3). An optimal solution could set X to 1, Y to 0, and Z to 0, gaining $3W - \varepsilon$. The proposed algorithm, however, could possibly set X to 0, earning only $2W$.

One could think of ordering the variables as it is done in the FMS Algorithm, privileging variables that exhibit a large difference of clause weights. However, in the example, X does exhibit the largest difference of clause weights. Still, setting X to 0 misses the optimal solution. The FMS Algorithm, in contrast, finds the optimal solution.

Appendix C Safe Variables

THEOREM 2: [*Safe Truth Values are Safe*]

Let \mathcal{X} be a set of variables, let v be a partial assignment on \mathcal{X} and let \mathcal{C} be a weighted set of clauses. Let x be a safe variable with safe truth value t . Let $v_o \supseteq v$ be an assignment that extends v and maximizes the sum of the weights of the satisfied clauses. Let v'_o be a variant of v_o that assigns t to x :

$$v'_o = v_o \setminus \{x \rightarrow 1, x \rightarrow 0\} \cup \{x \rightarrow t\}$$

Then

$$\sum_{c \in \mathcal{C} \text{ satisfied in } v'_o} w(c) \geq \sum_{c \in \mathcal{C} \text{ satisfied in } v_o} w(c)$$

Proof: Theorem 2 has first been proven for *unweighted* MAX SAT in [25] as the *Dominating Unit Clause Rule*. [44] provides a proof for Weighted MAX SAT with two literals per clause. The proof can be generalized to other cases where all clause have the same number of literals. We provide a shorter, general proof here.

Let \mathcal{X} be a set of variables, let v be a partial assignment on \mathcal{X} and let \mathcal{C} be a weighted set of clauses. Let x be a safe variable with safe truth value t (see Definition 10). Let $v_o \supseteq v$ be an assignment that extends v and maximizes the sum of the weights of the satisfied clauses. Let v'_o be a variant of v_o that assigns t to x :

$$v'_o = v_o \setminus \{x \rightarrow 1, x \rightarrow 0\} \cup \{x \rightarrow t\}$$

We have to prove

$$\sum_{c \in \mathcal{C} \text{ satisfied in } v'_o} w(c) \geq \sum_{c \in \mathcal{C} \text{ satisfied in } v_o} w(c)$$

If $v_o(x) = t$, it follows $v_o = v'_o$ and the claim follows immediately. Now assume $v_o(x) = \neg t$. We first observe that the clauses satisfied by v will be satisfied by both v_o and v'_o , because $v_o \supseteq v$ and $v'_o \supseteq v$. Hence, it suffices to consider only the clauses that are not satisfied in v . We define \mathcal{C}' to be the set of clauses that are not satisfied by v . Then we have to prove

$$\sum_{c \in \mathcal{C}' \text{ satisfied in } v'_o} w(c) \geq \sum_{c \in \mathcal{C}' \text{ satisfied in } v_o} w(c)$$

Since v_o differs from v'_o only in the assignment of x , the clauses that do not contain x will either be satisfied in both v_o and v'_o or in none of them:

$$\sum_{\substack{c \in \mathcal{C}' \text{ satisfied in } v'_o \\ x \notin c}} w(c) = \sum_{\substack{c \in \mathcal{C}' \text{ satisfied in } v_o \\ x \notin c}} w(c)$$

Hence, we have to prove only

$$\sum_{\substack{c \in \mathcal{C}' \text{ satisfied in } v'_o \\ x \in c}} w(c) \geq \sum_{\substack{c \in \mathcal{C}' \text{ satisfied in } v_o \\ x \in c}} w(c) \quad (*)$$

We consider the left-hand-side of (*) and compute a lower bound for it. Given that $v'_o(x) = t$, v'_o will satisfy at least the clauses in which x appears with polarity t :

$$\sum_{\substack{c \in \mathcal{C}' \text{ satisfied in } v'_o \\ x \in c}} w(c) \geq \sum_{\substack{c \in \mathcal{C}' \\ x \in c^t}} w(c)$$

Now we consider the right-hand-side of (*). v_o will satisfy all clauses that contain x with polarity $\neg t$. It cannot satisfy the clauses that contain x with polarity t and that were unit clauses in v . However, it could potentially satisfy the non-unit clauses that contain x with polarity t . Thus, we obtain the following upper bound on the right-hand-side of (*):

$$\sum_{\substack{c \in \mathcal{C}' \\ x \in c^{\neg t}}} w(c) + \sum_{\substack{c \in \mathcal{C}' \text{ non-unit in } v \\ x \in c^t}} w(c) \geq \sum_{\substack{c \in \mathcal{C}' \text{ satisfied in } v_o \\ x \in c}} w(c)$$

Using these two bounds in (*), we have to prove

$$\sum_{\substack{c \in \mathcal{C}' \\ x \in c^t}} w(c) \geq \sum_{\substack{c \in \mathcal{C}' \\ x \in c^{\neg t}}} w(c) + \sum_{\substack{c \in \mathcal{C}' \text{ non-unit in } v \\ x \in c^t}} w(c)$$

Subtracting the second summand on both sides yields

$$\sum_{\substack{c \in \mathcal{C}' \\ \text{unit clause in } v \\ x \in c^t}} w(c) \geq \sum_{\substack{c \in \mathcal{C}' \\ x \in c^{-t}}} w(c)$$

This is the definition of x being a safe variable with safe truth value t .

Asymmetry. Note the asymmetry in the rule for safe variables: To set a variable x to a truth value t , the rule considers the *unit clauses* where x appears with polarity t and compares them to the *unsatisfied clauses* where x appears with polarity $\neg t$. One might be tempted to compare just the unsatisfied clauses where x appears with polarity t to the unsatisfied clauses where x appears with polarity $\neg t$, no matter whether the clauses are unit clauses or not. This algorithm, however, performs worse than the FMS Algorithm in certain situations that are important in the SOFIE setting, see Section B.

Appendix D Approximation Guarantee of FMS*

THEOREM 3: [*Approximation Guarantee of the FMS* Algorithm*]
 The FMS* Algorithm has an approximation guarantee of 1/2.

Proof: We are given a Weighted MAX SAT problem in the form of a set \mathcal{X} of variables and a set \mathcal{C} of weighted clauses. We have to prove that the FMS* algorithm maintains an approximation guarantee of 1/2. As in the preceding proof in Appendix A for Theorem 1, we construct two sets $\mathcal{C}^- \subseteq \mathcal{C}$ and $\mathcal{C}^+ \subseteq \mathcal{C}$, which are updated after every step of the algorithm. In each step, the algorithm will assign a truth value t to a variable x (either by DUC Propagation or by the FMS Algorithm). As in the previous proof, we update the sets \mathcal{C}^- and \mathcal{C}^+ as follows before x is assigned:

$$\begin{aligned}\mathcal{C}^- &:= \mathcal{C}^- \cup \{c \mid c \in \mathcal{C} \text{ unsatisfied clause, } x \in c^{-t}\} \\ \mathcal{C}^+ &:= \mathcal{C}^+ \cup \{c \mid c \in \mathcal{C} \text{ unit clause, } x \in c^t\}\end{aligned}$$

The variable x can be assigned in two ways:

1. x can be assigned by the FMS Algorithm. Then,

$$\sum_{\substack{c \in \mathcal{C} \text{ unit clause} \\ x \in c^t}} w(c) \geq \sum_{\substack{c \in \mathcal{C} \text{ unit clause} \\ x \in c^{-t}}} w(c)$$

2. x can be assigned by DUC Propagation. Then, by the definition of DUC Propagation,

$$\sum_{\substack{c \in \mathcal{C} \text{ unit clause} \\ x \in c^t}} w(c) \geq \sum_{\substack{c \in \mathcal{C} \text{ unsatisfied} \\ x \in c^{-t}}} w(c)$$

Thus, in both cases

$$\sum_{\substack{c \in \mathcal{C} \\ x \in c^t}} w(c) \geq \sum_{\substack{c \in \mathcal{C} \\ x \in c^{-t}}} w(c)$$

Hence, the following invariance condition holds throughout the course of the algorithm:

$$\sum_{c \in \mathcal{C}^+} w(c) \geq \sum_{c \in \mathcal{C}^-} w(c)$$

As shown in Appendix A, this entails an approximation guarantee of $1/2$.

Bibliography

- [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-quantifying 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. Suchanek, M. Sozio, G. Weikum. Sofie: A Self-Organizing Framework for Information Extraction. Research Report MPI-I-2008-5-012, Max-Planck Institute for Informatics, 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.

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](ftp://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-2007-5-003	Fabian M. Suchanek, Gjergji Kasneci, Gerhard Weikum	Yago: A Large Ontology from Wikipedia and WordNet
MPI-I-2007-RG1-002	T. Hillenbrand, C. Weidenbach	Superposition for Finite Domains
MPI-I-2007-5-002	K. Berberich, S. Bedathur, T. Neumann, G. Weikum	A Time Machine for Text Search
MPI-I-2007-5-001	G. Kasneci, F.M. Suchanek, G. Ifrim, M. Ramanath, G. Weikum	NAGA: Searching and Ranking Knowledge
MPI-I-2007-4-006	C. Dyken, G. Ziegler, C. Theobalt, H. Seidel	GPU Marching Cubes on Shader Model 3.0 and 4.0
MPI-I-2007-4-005	T. Schultz, J. Weickert, H. Seidel	A Higher-Order Structure Tensor
MPI-I-2007-4-004	C. Stoll	A Volumetric Approach to Interactive Shape Editing
MPI-I-2007-4-003	R. Bargmann, V. Blanz, H. Seidel	A Nonlinear Viseme Model for Triphone-Based Speech Synthesis
MPI-I-2007-4-002	T. Langer, H. Seidel	Construction of Smooth Maps with Mean Value Coordinates
MPI-I-2007-4-001	J. Gall, B. Rosenhahn, H. Seidel	Clustered Stochastic Optimization for Object Recognition and Pose Estimation
MPI-I-2007-2-001	A. Podelski, S. Wagner	A Method and a Tool for Automatic Verification of Region Stability for Hybrid Systems
MPI-I-2007-1-002	E. Althaus, S. Canzar	A Lagrangian relaxation approach for the multiple sequence alignment problem
MPI-I-2007-1-001	E. Berberich, L. Kettner	Linear-Time Reordering in a Sweep-line Algorithm for Algebraic Curves Intersecting in a Common Point
MPI-I-2006-5-006	G. Kasnec, F.M. Suchanek, G. Weikum	Yago - A Core of Semantic Knowledge
MPI-I-2006-5-005	R. Angelova, S. Siersdorfer	A Neighborhood-Based Approach for Clustering of Linked Document Collections
MPI-I-2006-5-004	F. Suchanek, G. Ifrim, G. Weikum	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-2006-4-010	A. Belyaev, T. Langer, H. Seidel	Mean Value Coordinates for Arbitrary Spherical Polygons and Polyhedra in \mathbb{R}^3
MPI-I-2006-4-009	J. Gall, J. Potthoff, B. Rosenhahn, C. Schnoerr, H. Seidel	Interacting and Annealing Particle Filters: Mathematics and a Recipe for Applications
MPI-I-2006-4-008	I. Albrecht, M. Kipp, M. Neff, H. Seidel	Gesture Modeling and Animation by Imitation

MPI-I-2006-4-007	O. Schall, A. Belyaev, H. Seidel	Feature-preserving Non-local Denoising of Static and Time-varying Range Data
MPI-I-2006-4-006	C. Theobalt, N. Ahmed, H. Lensch, M. Magnor, H. Seidel	Enhanced Dynamic Reflectometry for Relightable Free-Viewpoint Video
MPI-I-2006-4-005	A. Belyaev, H. Seidel, S. Yoshizawa	Skeleton-driven Laplacian Mesh Deformations
MPI-I-2006-4-004	V. Havran, R. Herzog, H. Seidel	On Fast Construction of Spatial Hierarchies for Ray Tracing
MPI-I-2006-4-003	E. de Aguiar, R. Zayer, C. Theobalt, M. Magnor, H. Seidel	A Framework for Natural Animation of Digitized Models
MPI-I-2006-4-002	G. Ziegler, A. Tevs, C. Theobalt, H. Seidel	GPU Point List Generation through Histogram Pyramids
MPI-I-2006-4-001	A. Efremov, R. Mantiuk, K. Myszkowski, H. Seidel	Design and Evaluation of Backward Compatible High Dynamic Range Video Compression
MPI-I-2006-2-001	T. Wies, V. Kuncak, K. Zee, A. Podelski, M. Rinard	On Verifying Complex Properties using Symbolic Shape Analysis
MPI-I-2006-1-007	H. Bast, I. Weber, C.W. Mortensen	Output-Sensitive Autocompletion Search
MPI-I-2006-1-006	M. Kerber	Division-Free Computation of Subresultants Using Bezout Matrices
MPI-I-2006-1-005	A. Eigenwillig, L. Kettner, N. Wolpert	Snap Rounding of Bézier Curves
MPI-I-2006-1-004	S. Funke, S. Laue, R. Naujoks, L. Zvi	Power Assignment Problems in Wireless Communication
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 Empirical 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-2005-1-007	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