Semantic Bootstrapping: A Theoretical Perspective

Wentao Wu, Hongsong Li, Haixun Wang, and Kenny Q. Zhu

Abstract—Knowledge acquisition is an iterative process. Most previous work has focused on bootstrapping techniques based on syntactic patterns, that is, each iteration finds more syntactic patterns for subsequent extraction. However, syntactic bootstrapping is incapable of resolving the inherent ambiguities in the syntactic patterns. The precision of the extracted results is thus often poor. On the other hand, semantic bootstrapping bootstraps directly on knowledge rather than on syntactic patterns, that is, it uses existing knowledge to understand the text and acquire more knowledge. It has been shown that semantic bootstrapping can achieve superb precision while retaining good recall. Nonetheless, the working mechanism of semantic bootstrapping remains elusive. In this paper, we present a detailed analysis of semantic bootstrapping from a theoretical perspective. We show that the efficiency and effectiveness of semantic bootstrapping can be theoretically guaranteed. Our experimental evaluation results substantiate the theoretical analysis.

Index Terms—Algorithm, big data, information extraction, semantic bootstrapping

1 Introduction

The problem of extracting *isA* relations in the *open* domain has been studied for years. State-of-the-art systems, such as KnowItAll [1], TextRunner [2], and NELL [3], use a bootstrapping approach. They start with some seed examples and/or seed patterns of the target relations. They next look for occurrences of these seed examples in the corpus, and derive new patterns. They then use the new patterns to extract more instances of the relations. The iteration continues until no more new patterns are learned. In the rest of this paper, we refer to this idea as *syntactic bootstrapping*. Fig. 1 gives a canonical view of this framework.

The philosophy of syntactic bootstrapping is that, in order to find more relations, we need more syntactic patterns. However, this is often not true. One-to-one mapping between syntactic patterns and underlying *knowledge* (i.e., the pairs we are interested in) does not always exist. Sometimes one pattern can mean multiple things and multiple patterns can refer to the same thing. This disconnect between the patterns and knowledge means that acquiring more patterns does not always give us more knowledge, but rather ambiguity and noise [4].

The problem of ambiguity is ubiquitous in almost all syntactic patterns, even for those hand-crafted by linguistic experts. For instance, Table 1 lists the well-known Hearst patterns [5] used by nearly every existing information extraction system for the purpose of extracting *isA* relations. Now consider the following example sentences:

Example 1 (Ambiguity in Syntactic Patterns).

1) $\cdots$ animals other than dogs such as cats $\cdots$
2) $\cdots$ companies such as IBM, Nokia, Proctor and Gamble $\cdots$
3) $\cdots$ representatives in North America, Europe, the Middle East, Australia, Mexico, Brazil, Japan, China, and other countries $\cdots$
4) $\cdots$ classic movies such as Gone with the Wind $\cdots$

In these cases, patterns are incapable of making the right choices in the presence of ambiguities: 1) dogs will be incorrectly recognized as the super-concept[1]; 2) *Proctor* and *Gamble* are mistakenly extracted as two companies rather than one; 3) *North America*, *Europe*, and *the Middle East* will also be deemed as countries; 4) nothing can be extracted since *Gone with the Wind* is not a noun phrase that the pattern is looking for.

To address the ambiguity issue, syntactic bootstrapping approaches have to use more strict syntactic rules in their extractions, which often dramatically sacrifice recall. For instance, when extracting *isA* pairs, KnowItAll only focuses on sub-concepts that are *proper nouns* [1]. Unlike that, in [4] the authors outlined a conceptually different iterative framework, which bootstraps on knowledge rather than on syntactic patterns. We refer to this approach as *semantic bootstrapping*. It differs from syntactic bootstrapping in that it uses a fixed set of input patterns (e.g., the Hearst patterns) and relies on using existing knowledge (e.g., the pairs already extracted with their frequency) to understand more text and acquire more knowledge. As Fig. 2 depicts, in each round of

- *W. Wu is with Microsoft Research, Redmond, WA 98052. E-mail: wentwu@microsoft.com.*
- *H. Li is with Microsoft Research Asia, Beijing, 100080, China. E-mail: hongsli@microsoft.com.*
- *H. Wang is with Google Research, Mountain View, CA 94043. E-mail: haixun@google.com.*
- *K.Q. Zhu is with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China. E-mail: kzhu@cs.sjtu.edu.cn.*

Manuscript received 23 Nov. 2015; revised 1 Oct. 2016; accepted 8 Oct. 2016. Date of publication 19 Oct. 2016; date of current version 9 Jan. 2017. Recommended for acceptance by D. Barbosa. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TKDE.2016.2619347

1. *isA* relation is between super-concepts and sub-concepts. For example, in the relation "cat *isA* animal", "animal" is the super-concept and "cat" is the sub-concept. In Example 1, the underlined term is the super-concept, and the italicized terms are its sub-concepts. For a given *isA* relation (x, y), we also call x the concept and y the instance, although y may itself be a concept as well.

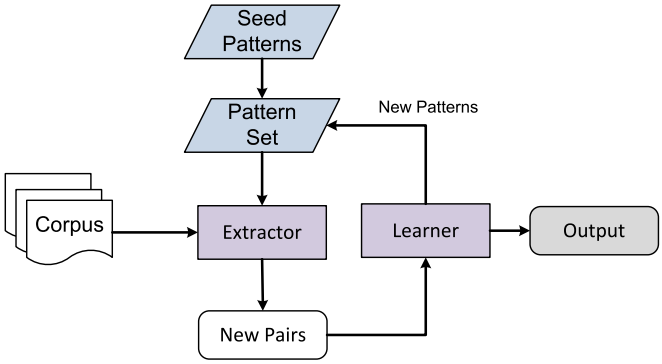


Fig. 1. Syntactic bootstrapping.

iteration, the extractor extracts new pairs with the help of the current knowledge, and then uses these new pairs to enrich the knowledge.[2] Albeit a simpler framework, this approach demonstrates exceptional strength in disambiguating otherwise unaccessible pairs and thus achieves superb precision while maintaining good recall in the extracted pairs.[3]

Nonetheless, the underlying working mechanism of semantic bootstrapping remains elusive in [4]: were the results reported just by chance? In this paper, we present a theoretical analysis as well as an extended experimental study to provide deeper insights into semantic bootstrapping. We show that the efficiency and effectiveness of semantic bootstrapping can be theoretically guaranteed. Specifically, the required number of iterations is $O(\log|\Gamma|)$, where Γ is the set of extracted pairs; and the precision of the extracted pairs is very close to that of the pairs extracted in the bootstrapping stage (i.e., the first two rounds of iterations), which are usually of high quality in practice. Our experimental evaluation results substantiate the theoretical analysis.

2 Semantic Bootstrapping

For self-containment purpose, in this section we first formulate the problem of extracting *isA* relations and then briefly describe the semantic bootstrapping framework. We refer the readers to [4] for more details of semantic bootstrapping.

2.1 Problem Formulation

isA relation can be extracted from sentences that match any of the Hearst patterns, e.g.,

"$\cdots$ in countries **such as** *China*, *Japan*, $\cdots$"

2. The "semantic" here might be a bit misleading. Our purpose is to distinguish our approach from bootstrapping procedures that aim for harvesting more and more syntactic patterns. The current form of "semantic" in our approach is rudimentary: we simply use statistics as the type of "semantics" or "knowledge." Nonetheless, it is not difficult to incorporate more "semantics" into our approach. For instance, we can add annotated *isA* relation pairs to help increase the accuracy of super-concept and sub-concept detection (see Algorithm 1).

3. Most errors in Example 1 are due to the fact that Hearst-like patterns ignore syntax and syntactic ambiguities. The patterns are flat and not formulated over a parse tree. This raises the question that, if no syntactic ambiguities would exist, would then semantic bootstrapping be obsolete? It is true that using more advanced syntactic techniques such as parser trees might help with some cases, e.g., sentence 1) in Example 1. However, It cannot address many other cases. For instance, consider sentence 3) in Example 1. It does not contain any syntactic ambiguity. Nonetheless, we would end up with incorrect extractions such as (*countries*, *the Middle East*) if we only followed syntactic approaches.

TABLE 1
The Hearst Patterns (*NP* Stands for *Noun Phrase*)

ID	Pattern
1	*NP* such as {*NP*,}*{(or | and)} *NP*
2	such *NP* as {*NP*,}*{(or | and)} *NP*
3	*NP*{,} including {*NP*,}*{(or | and)} *NP*
4	*NP*{,*NP*}*{,} and other *NP*
5	*NP*{,*NP*}*{,} or other *NP*
6	*NP*{,} especially {*NP*,}*{(or | and)} *NP*

Given such a sentence s, our goal is then to extract all pairs (x, y) in s such that "y *isA* x". For instance, from the above sentence, we want to extract (*country*, *China*) and (*country*, *Japan*). Formally, we can represent s with a triple:

$$s = (X_s, \langle P\rangle, Y_s),$$

where $X_s = \{x_1, \ldots, x_m\}$ is the set of all candidate super-concepts, $\langle P\rangle$ is the pattern keywords (e.g., the "*such as*" in the above example sentence), and $Y_s = \{y_1, \ldots, y_n\}$ is the set of all candidate sub-concepts. Ideally, we would like both $|X_s| = 1$ and $|Y_s| = 1$ so that there is no ambiguity. Unfortunately, in practice, this is rarely the case, and our goal is to identify those valid x's and y's among the candidates in X_s and Y_s. Here, naturally, we say that a pair (x, y) is valid if the relationship "y *isA* x" holds. If (x, y) is valid, then both x and y are valid.

2.2 Properties

The semantic bootstrapping framework relies on a couple of basic properties of the sentences that match the Hearst patterns to distinguish valid *isA* pairs from invalid ones.

Property 1. *For most of the sentences, there is one and only one valid* $x \in X_s$.

While in theory the mapping from valid $x \in X_s$ to valid $y \in Y_s$ could be many-to-many, in practice we find it is very unlikely that more than one x in X_s is valid. Intuitively, if more than one super-concept is valid, then s itself might be too ambiguous to be correctly parsed even by human beings. In fact, so far we have not even found such a highly ambiguous sentence in our corpus yet.

Property 2. *The closer a* $y \in Y_s$ *is to* $\langle P\rangle$, *the more likely that* y *is valid*.

Although it is arguable, we find that when enumerating sub-concepts, people tend to list those that they are familiar with first.

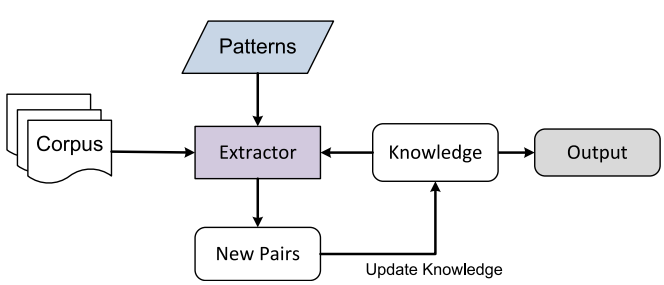


Fig. 2. Semantic bootstrapping.

Property 3. *If $y_k \in Y_s$ is valid, then most likely $y_1, \ldots, y_{k-1}$ are all valid.*

This means, there is usually a *boundary* in Y_s that delimits valid sub-concepts from invalid ones. For instance, in the sentence 3) of Example 1, the candidate "*Australia*" plays the role of a sentinel.

Remark 1. Note that the three properties here are not restricted to *isA* relation extraction. For example, the *authorship* relation can be extracted from sentences like "Victor Hugo **wrote** *The Hunchback of Notre-Dame* **and** *Les Misérables*."

In general, as long as we can identify the three components X_s, $\langle P \rangle$, and Y_s from a sentence s, the above properties are likely to hold and therefore we might be able to apply our semantic bootstrapping framework discussed next. Property 1 usually holds, partially due to the convention when people are writing a sentence. In English grammar, a sentence usually consists of three parts: the *subject*, the *predicate*, and the *object*. For most of the sentences, they only have one subject while they can have multiple objects. Properties 2 and 3 are usually valid as well, if the sentence contains objects that are chained by using conjunctions such as "and" or "or". In fact, the pattern "$y_1, \ldots,$ and/or y_n" occurs very frequently in English documents, and is well known as the *coordination pattern* in the literature [6].

Algorithm 1. *isA* Relation Extraction

Input: P, the Heast patterns; S, sentences that match any of the patterns in P
Output: Γ, the extracted *isA* pairs

1: $\Gamma \leftarrow \emptyset$;
2: $i \leftarrow 1$;
3: **while** *true* **do**
4: $\quad \Delta_i \leftarrow \emptyset$;
5: $\quad$ **foreach** $s \in S$ **do**
6: $\quad\quad X_s, Y_s \leftarrow ExtractCandidates(s)$;
7: $\quad\quad$ **if** $|X_s| > 1$ **then**
8: $\quad\quad\quad X_s \leftarrow DetectSuper(X_s, Y_s, \Gamma_{i-1})$;
9: $\quad\quad$ **if** $|X_s| = 1$ **then**
10: $\quad\quad\quad Y_s \leftarrow DetectSub(X_s, Y_s, \Gamma_{i-1})$;
11: $\quad\quad\quad$ add valid pairs to Δ_i;
12: $\quad\quad$ **end**
13: $\quad$ **end**
14: $\quad$ break if $\Delta_i = \emptyset$;
15: $\quad \Gamma_i \leftarrow \Gamma_{i-1} \cup \Delta_i$;
16: $\quad i \leftarrow i + 1$;
17: **end**
18: **return** Γ;

2.3 The Algorithm

Algorithm 1 outlines the method. Here, we use Γ to represent the *multiset* or *bag* of the pairs that we have discovered so far. We also use Γ_i to denote the Γ after the ith round of iteration in Algorithm 1, and use $\Delta_i = \Gamma_i - \Gamma_{i-1}$ to denote the multiset of pairs added in round i. Initially, $\Gamma_0 = \emptyset$. We define a count function $n(x, y)$ which returns how many times the pair (x, y) has been discovered in the corpus. Initially, Γ is empty. We search for candidate pairs in the text, and we use Γ to help identify valid ones among them. Specifically, we first call *ExtractCandidates* to extract candidate super-concepts and sub-concepts. The strategy in this stage is rather straightforward: we extract all noun phrases (up to the pattern keywords) as candidates for super-concepts, and we use ",", "and", and "or" as the delimiters for candidate sub-concepts. Next, if $|X_s| > 1$, we further call *DetectSuper* to determine the valid super-concept. Once the super-concept is chosen, we can then call *DetectSub* to detect valid sub-concepts. We will discuss some details of these two procedures shortly. Finally, we expand Γ by merging the newly discovered pairs (line 15). We keep iterating until we cannot extract any new pairs.

Remark 2. There is a subtle but nontrivial difference between Algorithm 1 and the one described in [4]. Note that, in Algorithm 1, new pairs identified in the current round of iteration are added into Γ all together at the end of the round (i.e., *lazy update*), while previously Γ was updated immediately when some new valid pair was identified (i.e., *eager update*). Eager update has the advantage of exploiting the new knowledge as soon as possible. As a result, it has the potential to identify more pairs in each round of iteration and therefore speed up the whole extraction procedure. However, it also has a subtle drawback that its results may depend on the order of the input sentences. This is fine if we can always enforce the same ordering, e.g., by sorting the sentences with respect to their unique identifiers or appending new sentences only to the end of the disk file that stores S. However, maintenance of S is then costly and might be prohibitive in practice for a frequently updated system like Probase [4]. By switching from eager update to lazy update, we can both get rid of the maintenance overhead and the dependency on the input order, although we sacrifice some efficiency by slightly increasing the number of rounds of iteration. Nonetheless, as we will see in Section 3, the efficiency of Algorithm 1 can be theoretically guaranteed. Moreover, we have observed very similar results regarding the precision of the extracted pairs (see Section 5).

2.4 Super-Concept Detection

In the case $|X_s| > 1$, we need to decide the correct super-concept of s. The basic idea is to compute the likelihood $p(x_i|Y_s)$ for each $x_i \in X_s$, and then pick the one with the maximum likelihood.

Consider $p(x_i|Y_s)$, we have

$$p(x_i|Y_s) \propto p(x_i)p(Y_s|x_i) \propto n(x_i) \prod_{j=1}^{n} p(y_j|x_i). \tag{1}$$

Here, we have assumed that the y_j's are independent given x_i. Furthermore,

$$p(y_j|x_i) = \frac{p(x_i, y_j)}{p(x_i)} = \frac{n(x_i, y_j)}{n(x_i)}, \tag{2}$$

where $n(x_i) = \sum_{j=1}^{n} n(x_i, y_j)$. However, if $(x_i, y_j) \notin \Gamma$, then $n(x_i, y_j) = 0$, which implies that a currently unseen pair will make $p(x_i|Y_s) = 0$. To overcome this issue, we use the well-known *additive smoothing* technique by refining $p(y_j|x_i)$ as

$$p(y_j|x_i) = \frac{n(x_i, y_j) + \alpha}{n(x_i) + n\alpha}, \tag{3}$$

where $\alpha > 0$ is the smoothing parameter. Therefore,

$$p(x_i|Y_s) \propto \frac{\prod_{j=1}^{n}(n(x_i, y_j) + \alpha)}{(n(x_i) + n\alpha)^{n-1}}. \quad (4)$$

Without loss of generality, let x_1 and x_2 be the candidates with the two *largest* likelihoods such that $p(x_1|Y_s) \geq p(x_2|Y_s)$. We pick x_1 as the output of *DetectSuper* if the ratio

$$r(x_1, x_2) = \frac{p(x_1|Y_s)}{p(x_2|Y_s)}, \quad (5)$$

is greater than some threshold.

2.5 Sub-Concept Detection

Assume that we have identified the super-concept $X_s = \{x\}$ from a sentence. The next task is to find its sub-concepts from Y_s. Based on Properties 2 and 3, the strategy is to first find the *largest* scope wherein candidate sub-concepts are all valid, and then address the ambiguity issues inside the scope by using a similar likelihood-based approach to the one used in super-concept detection. Specifically, we find the largest k such that the likelihood $p(y_k|x)$ is above a threshold. On the other hand, if we cannot find any y_k satisfying the condition, then we assume $k = 1$, provided that y_1 is not ambiguous (i.e., it does not contain "and" or "or").

Example 2 (Sub-Concept Detection). For specificity let us again consider sentence 3) in Example 1. In terms of the problem formulation as was presented in Section 2.1, we have $X_s = \{\text{countries}\}$ and $Y_s = \{\text{North America, Europe, the Middle East, Australia, Mexico, Brazil, Japan, China}\}$. We expect that (countries, Australia) has significantly higher likelihood than (countries, the Middle East). As a result, "Australia" serves as the boundary in Y_s, and *DetectSub* would extract "Australia," "Mexico," "Brazil," "Japan," and "China" from the sentence.

3 Efficiency

In this section, we analyze the efficiency of Algorithm 1. Since the total number of pairs we can extract from the corpus is finite, and in each round we only add new valid pairs extracted from the sentences into Γ, Algorithm 1 is guaranteed to terminate. The efficiency depends on the number of iterations it executes. In the following presentation, we say a sentence s in round i is *ambiguous* if $|X_s| > 1$ *after* applying *DetectSuper*, and *unambiguous* otherwise.

Before we proceed, we first redefine the notation S to be the set of sentences we finally extract *at least one* pair, not the set of all sentences in our corpus. In practice, it is likely that we cannot extract any pair from some sentences. For example, if we finally cannot determine the correct super-concept of a sentence, then we fail to extract anything from it. These sentences do not contribute anything to our results, so we exclude them from our analysis below.

Our analysis of the number of iterations is based on the analysis of expected number of pairs that can be extracted in each iteration. We find that the latter shrinks exponentially as the iteration proceeds, which therefore implies a logarithmic convergence speed of the extraction procedure. For convenience of reference, we summarize the key notation used in our analysis in Table 2. We start our analysis by presenting a basic property of *DetectSub*.

TABLE 2
Notation Used in the Analysis of Algorithm 1

Notation	Description
Γ_i	All pairs extracted *after* iteration i
Δ_i	The pairs extracted *in* iteration i
Ω_i	The remaining pairs in Γ after iteration i
$\bar{L}_i$	The average number of pairs that can be extracted from a sentence after iteration i
σ_i^x	The probability that some $y \in \Delta_i^x$ becomes a new boundary sub-concept
P_i	The precision of pairs in iteration i

3.1 A Basic Property of *DetectSub*

Consider an arbitrary sentence s in round $i+1$. Note that if s is ambiguous in round $i+1$, then we cannot extract any pair from it. Hence we only need to focus on the case when s is unambiguous. Let x be the super-concept of s, and $Y_s = \{y_1, \ldots, y_n\}$. Assume that $y_1, \ldots, y_j$ have been detected by *DetectSub* before round $i+1$. Then we can extract some pair(s) from s in round $i+1$ only if there is some k ($j < k \leq n$) such that y_k can be identified by *DetectSub*. Note that we seek the maximum k in *DetectSub*, and all y_{j+1} to y_k will be extracted once y_k is identified. Therefore, we refer to y_k as the current *boundary* sub-concept of s. We have the following property for y_k.

Lemma 1. $(x, y_k) \in \Delta_i$, *where* $\Delta_i = \Gamma_i - \Gamma_{i-1}$.

Proof. Let $p_i(y_k|x)$ be the value of the likelihood $p(y_k|x)$ that *DetectSub* is concerned with in round i. If $(x, y_k) \notin \Delta_i$, then $n(x, y_k)$ does not change between round i and $i+1$. Hence, $p_{i+1}(y_k|x) \leq p_i(y_k|x)$. Since we failed to extract (x, y_k) from s in round i only if $p_i(y_k|x) < \epsilon$, we must have $p_{i+1}(y_k|x) < \epsilon$ as well. Here ϵ is the threshold. Therefore, y_k cannot be identified by *DetectSub*, a contradiction. □

Lemma 1 basically states that, if y_k could be a boundary sub-concept, then its frequency (i.e., $n(x, y_k)$) must have been changed in round i (i.e., $(x, y_k) \in \Delta_i$). Therefore, when searching for a potential boundary sub-concept in round $i+1$, we can just focus on such y_k's. Based on this observation, we next analyze the expected number of pairs that can be extracted in each round.

3.2 The Expected Number of Extracted Pairs

We further break down our analysis here into two smaller steps. For a given sentence, we are interested in the following two questions:

- What is the likelihood that we can extract at least one pair from it? We call this likelihood the *chance of success*.
- What is the expected number of extracted pairs given that we can extract at least one pair from it? We call this expectation the *expected successes*.

We next analyze these two problems one by one.

3.2.1 The Chance of Success

Define Δ_i^x to be the set of pairs in Δ_i with x the super-concept. Based on Lemma 1, the possible boundary y_k can only come

from Δ_i^x. We assume that each distinct y in Δ_i^x is equally likely to be this y_k, with some probability θ ($0 < \theta < 1$). Of course, for different s, θ may be different. So here θ should be viewed as the average over all sentences. θ may also depend on the round number i. We give some further analysis here. Note that θ is the probability of the event E_y: "$y \in \Delta_i^x$ is a new boundary y_k in round $i+1$ for s". E_y occurs if and only if the following two events occur:

- E_y^1: $j < k \leq n$, i.e., the new boundary should bring in at least one new sub-concept.
- E_y^2: $p(y_k|x) \geq \epsilon$, i.e., the new boundary should pass the likelihood threshold ϵ.

Assuming the independence of E_y^1 and E_y^2, we have $\theta = p(E_y) = p(E_y^1)p(E_y^2)$. On one hand, as the iteration proceeds, $p(E_y^1)$ is expected to decrease, since it is more and more difficult to identify new sub-concepts from a sentence given that the total number of valid sub-concepts in a sentence is fixed. On the other hand, $p(E_y^2)$ is expected to increase, since Γ keeps on growing and the count of a valid pair is increasing as well. We thus treat θ as a constant.

For the ease of exposition, we formalize this assumption in the following:

Assumption 1. *Each distinct $y \in \Delta_i^x$ is independently and equally likely to be the boundary sub-concept y_k, with probability θ.*

We note here that this assumption may not be valid in reality. It is certainly possible that some sub-concepts are more likely than the others to be the boundary. However, Assumption 1 is reasonable if we do not have any *prior* knowledge on which sub-concepts are more likely, according to the principle of maximum entropy. On the other hand, if we do have such prior knowledge, we may replace this "uniform distribution" assumption by the real distribution, which is a possible extension to the current framework.

Let $D(\Delta_i^x)$ be the set of distinct y's in Δ_i^x. Consider the probability σ_i^x that some $y \in D(\Delta_i^x)$ becomes a new boundary of s. By Assumption 1, the probability that none of the y's in $D(\Delta_i^x)$ can be a new boundary is then $(1-\theta)^{|D(\Delta_i^x)|}$. Define $q = 1 - \theta$. We have

$$\sigma_i^x = 1 - (1-\theta)^{|D(\Delta_i^x)|} = 1 - q^{|D(\Delta_i^x)|}. \tag{6}$$

3.2.2 The Expected Successes

Suppose that we can identify some new boundary of s. Let $\bar{L}_i$ be the average number of pairs that can be extracted from a sentence after round i. Similar to Assumption 1, we have the following assumption:

Assumption 2. *Each of $y_{j+1}, \ldots, y_n$ is equally likely to be the y_k with the largest k determined by* DetectSub.

The expected number of pairs extracted from s in round $i+1$ is then

$$E[L_{i+1}^s] = \frac{1}{\bar{L}_i}(1 + \cdots + \bar{L}_i) = \frac{1}{2}(1 + \bar{L}_i). \tag{7}$$

Equation (7) may be worth some further explanation. Consider the following example:

Example 3. Let s be "x such as y_1, y_2, and y_3." Then $\bar{L}_0 = 3$, assuming that s is the only sentence in S. Note that $\bar{L}_0$ is unknown to *DetectSub*. So by Assumption 2, *DetectSub* may extract 1, 2, or 3 sub-concepts, depending on which of y_1, y_2, and y_3 is determined to be the boundary sub-concept. The expected number of extracted sub-concepts is therefore $E[L_1] = \frac{1}{3}(1+2+3) = 2$.

3.2.3 Putting It Together

We are now ready to compute the number of expected pairs extracted in each round. Let $\Omega_i = \Gamma - \Gamma_i$, which is the *remaining* pairs in Γ that can be extracted *after* round i. As before, let Ω_i^x be the subset of Ω_i containing pairs with x the super-concept. Our next result shows that the number of remaining pairs decreases exponentially as the iteration proceeds:

Lemma 2. $|\Omega_{i+1}^x| < \gamma_i^x |\Omega_i^x|$, *where* $\gamma_i^x = \frac{1}{2}(1 + q^{|D(\Delta_i^x)|})$.

Proof. By the definition of Ω_i, we have

$$|\Omega_i^x| = \bar{L}_i |S^x|, \tag{8}$$

where S^x is the subset of S containing sentences with x the super-concept. By Equation (7), the number of pairs we expect to extract in round $i+1$ is

$$E[L_{i+1}^x] = \frac{1}{2}(1 + \bar{L}_i)|S^x|. \tag{9}$$

Thus the expected number of remaining pairs after round $i+1$ is then

$$\begin{aligned} |\Omega_{i+1}^x| &= |\Omega_i^x| - E[L_{i+1}^x] \cdot \sigma_i^x \\ &= \bar{L}_i |S^x| - \frac{1}{2}(1 + \bar{L}_i)|S^x| \cdot \sigma_i^x \\ &= \left[\left(1 - \frac{1}{2}\sigma_i^x\right)\bar{L}_i - \frac{1}{2}\sigma_i^x\right]|S^x| \\ &< \left(1 - \frac{1}{2}\sigma_i^x\right)\bar{L}_i |S^x|. \end{aligned} \tag{10}$$

Hence we now have

$$|\Omega_{i+1}^x| < \left(1 - \frac{1}{2}\sigma_i^x\right)|\Omega_i^x| = \frac{1}{2}(1 + q^{|D(\Delta_i^x)|})|\Omega_i^x|, \tag{11}$$

which completes the proof of the lemma. □

Remark 3. We have two remarks here for Lemma 2. First, for a particular x, the specific value of γ_i^x depends on the number of pairs in Δ_i^x. Since $q < 1$, $q^{|D(\Delta_i^x)|}$ can be ignored even for a very small Δ_i^x with only dozens of distinct pairs. Therefore, we can actually expect $|\Omega_{i+1}^x| < \frac{1}{2}|\Omega_i^x|$ for most of the rounds except for the last few ones. Thus, usually we can grab more than half of the remaining pairs in each round. Second, different x has different $|D(\Delta_i^x)|$, and hence has different γ_i^x. As discussed above, $\gamma_i^x \approx \frac{1}{2}$ unless $|D(\Delta_i^x)|$ is very small. In practice, it is well known that most pairs in Ω_i are from a small number of x's. These x's usually have large $D(\Delta_i^x)$. Since $|\Omega_{i+1}^x| < \frac{1}{2}|\Omega_i^x|$ for these x's, we can also expect $|\Omega_{i+1}| < \frac{1}{2}|\Omega_i|$ (except for the last several rounds).

3.3 The Convergence Rate of Algorithm 1

It is now easy to prove the following theorem, which basically states that Algorithm 1 will converge in logarithmic rate with respect to $|\Gamma|$. This is a natural corollary from Lemma 2.

Theorem 1. *Algorithm 1 is expected to end after* $\lceil \log_{\frac{1}{\gamma}} |\Gamma| \rceil + 1$ *rounds of iteration, where* $\gamma = \frac{1}{2}(1+q)$.

Proof. Based on Lemma 2,

$$|\Omega_{i+1}^x| < \frac{1}{2}(1 + q^{|D(\Delta_i^x)|})|\Omega_i^x|, \tag{12}$$

for a specific x. Since $q < 1$ and $|D(\Delta_i^x)| \geq 1$, we have

$$|\Omega_{i+1}^x| < \frac{1}{2}(1+q)|\Omega_i^x| = \gamma|\Omega_i^x|. \tag{13}$$

Note that this is true regardless of the x. Therefore,

$$|\Omega_{i+1}| < \frac{1}{2}(1+q)|\Omega_i|, \tag{14}$$

since $|\Omega_i| = \sum_x |\Omega_i^x|$. After N rounds of iteration, the number of remaining pairs that can be extracted is

$$|\Omega_N| < \gamma^{N-1}|\Omega_1| < \gamma^{N-1}|\Gamma|. \tag{15}$$

Since $q < 1$, we must have $\gamma = \frac{1}{2}(1+q) < 1$. Letting $\gamma^{N-1}|\Gamma| < 1$ gives us that $N > \log_{\frac{1}{\gamma}}|\Gamma| + 1$. This means, after $N' = \lceil \log_{\frac{1}{\gamma}}|\Gamma| \rceil + 1$ rounds, it is expected that $|\Omega_{N'}| < 1$. Algorithm 1 cannot extract more pairs and hence terminates. □

4 Precision

In this section, we analyze the precision of the pairs extracted by Algorithm 1. Since precision can only be manually evaluated, our goal here is not to give an explicit number. Rather, we develop a lower bound of the expected overall precision given that the precision of the first several rounds is known. Specifically, our analysis shows that the overall precision only depends on the precision of the pairs extracted in the first two rounds. Since in practice the precision of these pairs is usually very high, we can therefore expect high precision of all the pairs extracted.

In the following, we start by giving a representation of the total number of extracted pairs (i.e., $|\Gamma|$). Our analysis shows that $|\Gamma|$ can be approximately expressed in terms of the number of pairs extracted in the first two iterations. We then conduct a similar study on the total number of *correct* pairs extracted, which again connects it with the number of *correct* pairs extracted in the first two iterations. As a final step, we develop an expression for the overall precision based on the previous two results, and further derive a natural lower bound according to the established expression. Again, we refer the readers to Table 2 for the key notation used in this section.

4.1 The Total Number of Extracted Pairs

We first analyze the relationships between the $|\Delta_i^x|$'s, i.e., the number of pairs extracted in each round with x the super-concept. We have the following lemma:

Lemma 3. *Let* N *be the number of rounds Algorithm 1 iterates before its termination. Then*

$$\sum_{j=1}^{N} |\Delta_j^x| = |\Delta_1^x| + \frac{2}{1 - q^{|D(\Delta_1^x)|}}|\Delta_2^x| - |S^x|. \tag{16}$$

Proof. In the proof of Lemma 2, we have shown

$$|\Omega_{i+1}^x| = \left[\left(1 - \frac{1}{2}\sigma_i^x\right)\bar{L}_i - \frac{1}{2}\sigma_i^x\right]|S^x|. \tag{17}$$

Since $|\Omega_i^x| = \bar{L}_i|S^x|$, the above can be rewritten as

$$|\Omega_{i+1}^x| = \left(1 - \frac{1}{2}\sigma_i^x\right)|\Omega_i^x| - \frac{1}{2}\sigma_i^x|S^x|. \tag{18}$$

Since $|\Delta_{i+1}^x| = |\Omega_i^x| - |\Omega_{i+1}^x|$, it follows that

$$|\Delta_{i+1}^x| = \frac{1}{2}\sigma_i^x|\Omega_i^x| + \frac{1}{2}\sigma_i^x|S^x|. \tag{19}$$

Note that $|\Omega_i^x| = \sum_{j=i+1}^{N} |\Delta_j^x|$, which gives

$$|\Delta_{i+1}^x| = \frac{1}{2}\sigma_i^x \sum_{j=i+1}^{N} |\Delta_j^x| + \frac{1}{2}\sigma_i^x|S^x|, \tag{20}$$

or equivalently,

$$\left(1 - \frac{1}{2}\sigma_i^x\right)|\Delta_{i+1}^x| = \frac{1}{2}\sigma_i^x \sum_{j=i+2}^{N} |\Delta_j^x| + \frac{1}{2}\sigma_i^x|S^x|. \tag{21}$$

Therefore, we have

$$\sum_{j=i+2}^{N} |\Delta_j^x| = \frac{2 - \sigma_i^x}{\sigma_i^x}|\Delta_{i+1}^x| - |S^x|. \tag{22}$$

Letting $i = 1$ in the above equation gives

$$\sum_{j=3}^{N} |\Delta_j^x| = \frac{2 - \sigma_1^x}{\sigma_1^x}|\Delta_2^x| - |S^x|. \tag{23}$$

Since

$$\sum_{j=1}^{N} |\Delta_j^x| = |\Delta_1^x| + |\Delta_2^x| + \sum_{j=3}^{N} |\Delta_x^j|, \tag{24}$$

we have

$$\sum_{j=1}^{N} |\Delta_j^x| = |\Delta_1^x| + \frac{2}{\sigma_1^x}|\Delta_2^x| - |S^x|. \tag{25}$$

The lemma follows by noting $\sigma_1^x = 1 - q^{|D(\Delta_1^x)|}$. □

Since $|D(\Delta_1^x)|$ is usually large, we can expect that $1 - q^{|D(\Delta_1^x)|} \approx 1$. Hence, based on Lemma 3, we can have the following equation

$$|\Gamma^x| = \sum_{j=1}^{N} |\Delta_j^x| \approx |\Delta_1^x| + 2|\Delta_2^x| - |S^x|. \tag{26}$$

Here, following our notational convention, Γ^x represents the total number of extracted pairs with x the super-concept. Therefore, since $|\Gamma| = \sum_x |\Gamma^x|$, we have

$$\begin{aligned} |\Gamma| &= \sum_x |\Delta_1^x| + 2\sum_x |\Delta_2^x| - \sum_x |S^x| \\ &= |\Delta_1| + 2|\Delta_2| - |S|. \end{aligned} \tag{27}$$

The above analysis suggests that the total number of extracted pairs (i.e., $|\Gamma|$) can be expressed in terms of only the number of pairs extracted in the first two iteration. While this might seem surprising at first glance, it resonates with the fact that the number of pairs that can be extracted decreases exponentially (Lemma 2). In the following we will see that the total number of correct pairs extracted exhibits a very similar property.

4.2 The Total Number of Correct Pairs

We next analyze the number of correct pairs extracted by Algorithm 1. In the following discussion, we use Γ', Δ', and Ω' to denote the subsets of correct pairs in Γ, Δ, and Ω, respectively. We have

Lemma 4. *Let N be the number of rounds Algorithm 1 iterates before its termination. Then*

$$\sum_{j=1}^{N} |(\Delta_j^x)'| = |(\Delta_1^x)'| + \frac{2}{1-q^{|D(\Delta_1^x)|}} |(\Delta_2^x)'|. \tag{28}$$

Proof. The proof is almost the same as the proof of Lemma 3, with only one difference. Recall that, if we can extract at least one pair from a sentence in round $i+1$, then the expected number of pairs we can extract is $\frac{1}{2}(1+\bar{L}_i)$, where $\bar{L}_i$ is the average number of pairs that can be extracted from a sentence *after* round i (see Equation (7)). Now since we only focus on the correct pairs, let $\bar{L}'_i$ be the average number of *correct* pairs we can extract from a sentence after round i. Note that it is possible that we extract some pairs from a sentence but none of them is correct. So the expected number of correct pairs we can extract in round $i+1$ is

$$\frac{1}{\bar{L}'_i+1}(0+1+\cdots+\bar{L}'_i) = \frac{1}{2}\bar{L}'_i, \tag{29}$$

not $\frac{1}{2}(1+\bar{L}'_i)$. Therefore, following the same analysis as that in the proof of Lemma 2, we have

$$\begin{aligned} |(\Omega_{i+1}^x)'| &= |(\Omega_i^x)'| - \frac{1}{2}\bar{L}'_i \cdot \sigma_i^x |S^x| \\ &= \bar{L}'_i |S^x| - \frac{1}{2}\bar{L}'_i \cdot \sigma_i^x |S^x| \\ &= \left(1 - \frac{1}{2}\sigma_i^x\right)\bar{L}'_i |S^x|. \end{aligned} \tag{30}$$

Since $|(\Omega_i^x)'| = \bar{L}'_i |S^x|$, it follows that

$$\left|(\Omega_{i+1}^x)'\right| = \left[1 - \frac{1}{2}\sigma_i^x\right]\left|(\Omega_i^x)'\right|. \tag{31}$$

The rest of the proof is the same as that in Lemma 3, and hence we omit it here. □

Similarly, we expect that $1 - q^{|D(\Delta_1^x)|} \approx 1$. Hence

$$\begin{aligned} |\Gamma'| &\approx \sum_x |(\Delta_1^x)'| + 2\sum_x |(\Delta_2^x)'| \\ &= |\Delta'_1| + 2|\Delta'_2|. \end{aligned} \tag{32}$$

This suggests that the total number of correct pairs can also be expressed in terms of only the number of correct pairs extracted in the first two iterations. We are now ready to give an expression for the overall precision based on the results we have derived.

4.3 A Lower Bound for Overall Precision

Now, let P_1 and P_2 be the precision of Δ_1 and Δ_2, i.e., $P_1 = \frac{|\Delta'_1|}{|\Delta_1|}$ and $P_2 = \frac{|\Delta'_2|}{|\Delta_2|}$.

Theorem 2. *The precision P of Γ is*

$$P = \frac{|\Gamma'|}{|\Gamma|} = \frac{\alpha P_1 + 2P_2}{\alpha + 2 - \beta}, \tag{33}$$

where $\alpha = \frac{|\Delta_1|}{|\Delta_2|}$ and $\beta = \frac{|S|}{|\Delta_2|}$. Since $\alpha \geq 0$ and $\beta \geq 0$, it follows that

$$P \geq \frac{2}{2+\alpha} P_2. \tag{34}$$

Proof. The proof is by direct computation:

$$\begin{aligned} P = \frac{|\Gamma'|}{|\Gamma|} &= \frac{|\Delta'_1| + 2|\Delta'_2|}{|\Delta_1| + 2|\Delta_2| - |S|} \\ &= \frac{P_1|\Delta_1| + 2P_2|\Delta_2|}{|\Delta_1| + 2|\Delta_2| - |S|} = \frac{\alpha P_1 + 2P_2}{\alpha + 2 - \beta}, \end{aligned} \tag{35}$$

since by definition, $|\Delta'_1| = P_1|\Delta_1|$, and $|\Delta'_2| = P_2|\Delta_2|$. Furthermore, we have

$$P = \frac{P_1\alpha + 2P_2}{\alpha + 2 - \beta} \geq \frac{P_1\alpha + 2P_2}{\alpha + 2} \geq \frac{2}{2+\alpha} P_2, \tag{36}$$

since $\alpha \geq 0$ and $\beta \geq 0$. □

Theorem 2 implies a lower bound of P that only depends on α and P_2. Since $0 \leq \alpha \leq 1$, we then have

$$P \geq \frac{2}{3} P_2, \tag{37}$$

regardless of α. In practice, α is usually quite small since $|\Delta_2|$ is usually much larger than $|\Delta_1|$, for Δ_1 only serves the purpose of providing *seed* pairs for semantic bootstrapping. Therefore, we can expect that the lower bound is very close to P_2.

To better understand the lower bound for the overall precision, let us consider a concrete example:

Example 4 (Lower Bound for Precision). Suppose that we are given the following contrived corpus:

1) A and B such as C, D and E.
2) A and B such as C, D and E.
3) A and B such as C, E and D.
4) A and B such as D.
5) A such as C.
6) A such as C.
7) A such as D.
8) A such as D.

Assume that $\texttt{isA}(A, C)$ and $\texttt{isA}(A, E)$ is correct, but $\texttt{isA}(A, D)$ is incorrect. Furthermore, for simplicity assume that the threshold used by *DetectSub* for $n(x, y)$ is 1, i.e., the appearance of a sub-concept makes it eligible for the boundary. Algorithm 1 will then finish in three iterations:

I1) It will extract (from sentences 5 to 8): (A, C) twice and (A, D) twice. So the precision (P_1) is

$$P_1 = \frac{2}{2+2} = \frac{1}{2}.$$

I2) As (A, C) and (A, D) are much more likely than (B, C) and (B, D), A will be determined as the super-concept of sentences 1 to 4. Moreover, D now serves as the boundary sub-concept for sentences 1 to 4. Algorithm 1 will then extract (A, C), (A, D) from sentences 1 and 2, extract (A, C), (A, E), (A, D) from sentence 3, and extract (A, D) from sentence 4. As a result, it will extract (A, C) three times, (A, D) four times, and (A, E) once. So the precision (P_2) is

$$P_2 = \frac{3+1}{3+4+1} = \frac{1}{2}.$$

I3) E now serves as the boundary concept for sentences 1 and 2. Algorithm 1 will then extract (A, E) twice, once for each of sentences 1 and 2. So the precision (P_3) is

$$P_3 = \frac{2}{2} = 1.$$

In total, Algorithm 1 will extract (A, C) five times, (A, D) six times, and (A, E) three times. So the overall precision (P) is then

$$P = \frac{5+3}{5+6+3} = \frac{4}{7}.$$

On the other hand, the lower bound suggested by Theorem 2 is

$$\frac{2}{3}P_2 = \frac{2}{3} \cdot \frac{1}{2} = \frac{1}{3}.$$

Hence, it holds that $P \geq \frac{2}{3}P_2$.

We want to emphasize here that this example is only for illustrative purpose. The analysis presented in this section relies on the assumption that the corpus is large, otherwise some approximations (especially $1 - q^{|D(\Delta_1^x)|} \approx 1$ as was in Lemmas 3 and 4) may not work on small corpus.

5 Evaluation

We report our experimental evaluation results in this section. Our corpus contains more than 7 billion Web pages, which is 3.4 times larger than that used in [4].

5.1 Efficiency

We extracted overall 102,309,829 *isA* pairs. We further studied the number of pairs extracted in each round of iteration. Table 3 shows the number of pairs extracted ($|\Delta_i|$) and the number of remaining pairs ($|\Omega_i|$) for each round i, respectively. Note that, since Δ_i is by definition a multiset, we also report the number of distinct elements it contains ($|D(\Delta_i)|$), which are the *new* pairs extracted in round i.

We observe from Table 3 that $|\Omega_i|$ decreases exponentially, as predicted by Theorem 1. Moreover, the remaining number of pairs from the current round i is usually no more than half of that from the previous round $i-1$. As analyzed in Section 3 (see the remarks after Lemma 2), the upper bound of the shrinking factor γ should be close to $\frac{1}{2}$ in practice, due to the large $|\Delta_i|$ observed in each round.

TABLE 3
The Number of *isA* Pairs Extracted

Round i	$\lvert\Delta_i\rvert$	$\lvert D(\Delta_i)\rvert$	$\lvert\Omega_i\rvert$
1	26,492,477	16,736,068	321,276,392
2	244,880,870	56,060,246	76,395,522
3	48,582,780	17,515,818	27,812,742
4	16,214,502	7,060,475	11,598,240
5	7,069,892	2,907,529	4,528,348
6	2,204,261	1,047,007	2,324,087
7	1,619,613	567,942	704,474
8	523,076	286,324	181,398
9	106,641	73,762	74,757
10	51,644	39,520	23,113
11	23,113	15,138	0

We also notice that the first several rounds extract a dominant number of pairs, compared with the remaining rounds. However, this does not mean the later rounds are not useful. They are still important because: i) they improve the recall; and ii) they capture those *isA* pairs that involve concepts and instances in the *long tail* (i.e., concepts and instances that are not frequently mentioned in the corpus), which have been demonstrated to be useful in various applications [7], [8], [9], [10]. Nonetheless, here we might have exaggerated the importance of the long-tail concepts and entities if the users are actually not interested in them. In such cases, one could stop the iteration much earlier. In this sense, our algorithm provides some tunable trade-off between the amount of information it can extract and the running time it needs. A reasonable practice might be to terminate when the number of pairs extracted in the last iteration is below some threshold.

5.2 Precision

To estimate the correctness of the extracted *isA* pairs, we used the same benchmark as that in [4] with 40 concepts covering various domains (see Table 4). For each concept, to estimate its precision, we followed the same approach as in [4] by *randomly* picking 50 instances and manually evaluating their correctness. The average precision of the *isA* pairs over the benchmark concepts is 92.9 percent, which is very close to that reported in [4]. Fig. 3 further presents the precision of each individual concept.

In Table 5, we further examined the precision of the pairs extracted in round i (P_i) and the overall precision of the pairs extracted from round 1 to round i (Q_i). Here P_i and Q_i are evaluated based on the duplicated pairs (i.e., Δ_i). We notice that the overall precision when the iteration ends (i.e., $Q_{11} = 93.95$ percent) is higher than the 92.9 percent precision reported above, because the 92.9 percent precision was based on unique pairs. (Due to the high precision, double-counting the duplicates can increase the overall precision.)

Notice that we again have some trade-off between precision and recall here. As suggested by Table 5, P_i drops as the iteration proceeds while we indeed extract more valid pairs. Again, the trade-off depends on how the information

TABLE 4
Benchmark Concepts

Concept (# of Instances)	Representative Instances
actor (20226)	Tom Hanks, Marlon Brando, George Clooney
aircraft model (160)	Airbus A320-200, Piper PA-32, Beech-18
airline (3929)	British Airways, Deltae
airport (3299)	Heathrow, Gatwick, Stansted
album (8377)	Thriller, Big Calm, Dirty Mind
architect (2825)	Frank Gehry, Le Corbusier, Zaha Hadid
artist (179724)	Picasso, Bob Dylan, Madonna
book (63868)	Bible, Harry Potter, Treasure Island
cancer center (110)	Fox Chase, Care Alliance, Dana-Farber
celebrity (32220)	Madonna, Paris Hilton, Angelina Jolie
chemical compound (339)	carbon dioxide, phenanthrene, carbon monoxide
city (48357)	New York, Chicago, Los Angeles
company (356136)	IBM, Microsoft, Google
digital camera (802)	Canon, Nikon, Olympus
disease (58017)	AIDS, Alzheimer, chlamydia
drug (32557)	tobacco, heroin, alcohol
festival (13252)	Sundance, Christmas, Diwali
file format (3547)	PDF, JPEG, TIFF
film (73897)	Blade Runner, Star Wars, Clueless
food (69836)	beef, dairy, French fries
football team (442)	Real Madrid, AC Milan, Manchester United
game publisher (337)	Electronic Arts, Ubisoft, Eidos
internet protocol (479)	HTTP, FTP, SMTP
mountain (1977)	Everest, the Alps, the Himalayas
museum (7314)	the Louvre, Smithsonian, the Guggenheim
olympic sport (331)	gymnastics, athletics, cycling
operating system (6164)	Linux, Solaris, Microsoft Windows
political party (2012)	NLD, ANC, Awami League
politician (6065)	Barack Obama, Bush, Tony Blair
programming language (2472)	Java, Perl, PHP
public library (153)	Haringey, Calcutta, Norwich
religion (3830)	Christianity, Islam, Buddhism
restaurant (28651)	Burger King, Red Lobster, McDonalds
river (6379)	Mississippi,the Nile, Ganges
skyscraper (132)	the Empire State Building, the Sears Tower, Burj Dubai
tennis player (281)	Maria Sharapova, Andre Agassi, Roger Federer
theater (4174)	Metro, Pacific Place, Criterion
university (9954)	Harvard, Stanford, Yale
web browser (1182)	Internet Explorer, Firefox, Safari
website (34539)	YouTube, Facebook, MySpace

is utilized by the users. If precision is more important, we can stop the iteration earlier, while we can iterate for more rounds if recall is crucial. Nonetheless, a nice property of our framework is that the overall precision Q_i is guaranteed to be good by the time the algorithm converges.

Finally, we find that the overall precision shown in Table 5 matches our lower bound developed in Section 4 quite well. According to Theorem 2, the overall precision

TABLE 5
Precision of the Pairs Extracted

Round i	$\lvert\Delta_i\rvert$	P_i	Q_i
1	26,492,477	0.9728	0.9728
2	244,880,870	0.9713	0.9714
3	48,582,780	0.8877	0.9587
4	16,214,502	0.7976	0.9509
5	7,069,892	0.6846	0.9454
6	2,204,261	0.5403	0.9429
7	1,619,613	0.4378	0.9405
8	523,076	0.5	0.9398
9	106,641	0.3983	0.9397
10	51,644	0.3576	0.9396
11	23,113	0.3049	0.9395

$P \geq \frac{2}{2+\alpha} P_2$. According to Table 5, we have $\alpha = \frac{|\Delta_1|}{|\Delta_2|} \approx 0.1082$. Hence, the predicted $P \geq 0.9487 P_2 \approx 0.9215$, which is very close to the actual overall precision observed (i.e., $Q_{11} = 0.9395$).

5.3 Recall

Evaluation of recall is challenging. Specifically, the recall is defined as

$$recall = \frac{\#\text{ of valid isA pairs extracted}}{\#\text{ of valid isA pairs in the corpus}}. \quad (38)$$

Although computing the numerator is straightforward given that we have already evaluated the precision, estimating the denominator is an extremely hard problem. If we just count the number of distinct valid pairs from some sample sentences and try to scale up the estimate based on that, we cannot make the estimate reasonably accurate unless the sample size is sufficiently large. In fact, this problem of estimating the number of distinct values in a population is well known as "*estimating the number of species*" in statistical literature [11], and it has been shown that the estimation error cannot be bounded unless the sample size is $O(n)$, where n is the population size [12]. Since our corpus contains billions of sentences, it means that we need to manually identify valid pairs from billions of sample sentences, which is clearly prohibitive. Perhaps because of this difficulty, so far we are not aware of any previous work on open-domain information extraction that reported recall as defined by Equation (38).

Nonetheless, we further conducted experiments to evaluate "recall" based on the idea of sampling random sentences from the corpus. We randomly sampled 400 sentences from our corpus that the algorithm extracted at least one pair, and manually identified the correct super-concepts and

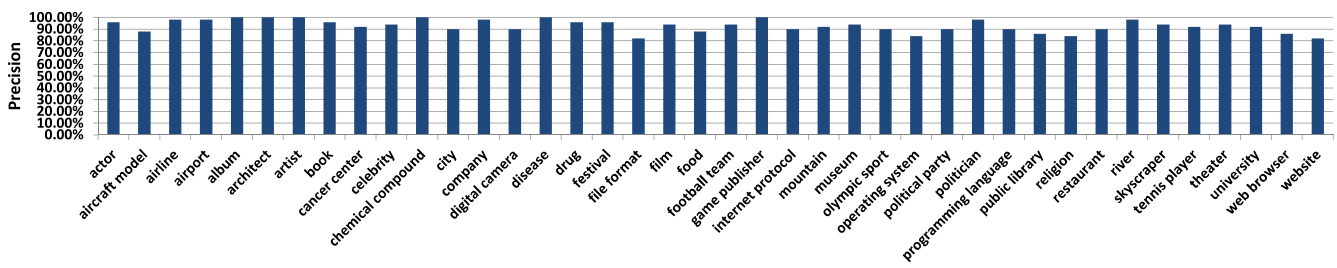


Fig. 3. Precision of randomly picked 50 instances over the benchmark concepts.

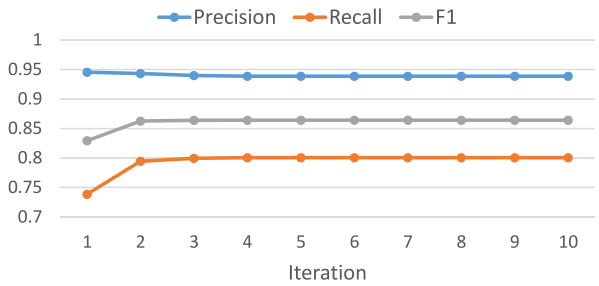


Fig. 4. Precision, recall, and F1 score over 400 random sample sentences.

sub-concepts. We then measured the precision and recall based on the extracted pairs and the ground truth.

Fig. 4 presents the results (as well as the F1 scores). We observe that, while the algorithm can achieve high precision (close to 94 percent when it terminates), the "recall" is relatively low (around 80 percent, which gives an F1 score of 86 percent). Note that, here the "recall" only means recall on this sample corpus, and we cannot make any inference on the recall over the whole corpus because of the aforementioned hardness result. However, in our experiments we do find that the algorithm does not perform well on rare sentences that contain many sub-concepts, usually because the algorithm cannot find enough evidence (i.e., sufficiently high frequency) for a boundary sub-concept that is close to the end of the sub-concept list. In this sense, the current version of *DetectSub* is a bit conservative. We leave improving recall as one of the main directions for future work.

6 Discussion and Critiques

In this section, we would like to discuss and give some critiques to the applicability and extensibility of the analysis presented in this paper.

First, in our analysis, we have employed several assumptions, some of which may not always hold in practice. The goal of the theoretical analysis in this paper is not to capture all real-world subtleties: such a model might be too complicated to be tractable for a formal analysis. For example, when analyzing the efficiency of Algorithm 1, rather than assuming a uniform distribution (Assumption 1), we could instead assume that the probability of a $y \in \Delta_i^x$ being the y_k follows some distribution $\Pr(y)$. But then we have to assume $\Pr(y)$ be certain known distribution (e.g., Gaussian) to make any further inference, which might still be unrealistic. Hence, rather than developing sophisticated models which might provide better insights but also be difficult to understand, our intention in this paper is to provide a simple, basic model and show what we can infer by just using this model. Our evaluation on a real data set shows that the results we derive from the model match the experimental observations, which substantiates the usefulness of the model. Of course, it remains interesting to further explore possible extensions to the current model to make it more general.

Second, while the analysis in the paper is dedicated to the approach in [4], it may also be applied to other cases. Note that the approach in [4] can be used to extract other types of relations (e.g., "part-of" relations [13]), provided that there are patterns bearing the properties stated in Section 2.2. Perhaps more importantly, similar analysis may be done for other bootstrapping procedures, which are popular in modern information extraction systems. Indeed, this is an essential motivation of this paper, which, we hope, provides an example and makes a first step towards this direction. However, Algorithm 1 is not really a prototypical algorithm used in "syntactic bootstrapping" approaches. Typically, some candidate scoring functions are used to decide if an extraction is correct [14]. The idea is to harvest more tuples with the current patterns, and expand the patterns by using the new tuples found. This is essentially more complicated compared with semantic bootstrapping analyzed in this paper. Because the set of patterns can change as syntactic bootstrapping proceeds, it is expected that there are more rounds of iterations. Meanwhile, the overall precision does not only depend on the precision of the tuples in the bootstrapping phase: it also depends on the quality of the patterns extracted (and vice versa) in later rounds. It is very interesting future work to see if the current framework we used for the analysis of semantic bootstrapping can be extended to analyze syntactic bootstrapping.

Third, one might also have the question regarding the implication and usefulness of such theoretical analysis. The meaning is that it will guide people on how to use semantic bootstrapping in practice. For example, our analysis suggests that we can trade off between precision and recall by using different iterations. As another example, as suggested by Theorem 2, the overall precision highly depends on the precision of the first two iterations. Therefore, to improve the quality of the extracted pairs, one should focus on improving the quality of the pairs extracted in the first two iterations. For instance, one may want to choose larger thresholds (used by *DetectSuper* and *DetectSub*) in the first two iterations and smaller thresholds in later iterations. As an alternative, one may also incorporate external, clean knowledge (e.g., *isA* relations from *WordNet* [15] or *Freebase* [16]) into the first two iterations to boost the precision.

7 Related Work

Automatically extracting relations from text corpus has been studied for decades. Early work focused on information extraction from a *closed* domain (e.g., news). These systems usually used supervised learning approaches such as Hidden Markov Models (e.g., [17]), rule learning (e.g., [18]), or Conditional Random Fields (e.g. [19]), which were elaborately tuned for that specific domain. As a result, while these approaches might be effective on documents that are similar to those in the training corpus, it is difficult to apply them to *open* domains like the whole Web, where the corpus and the target relations are much more diverse.

To the best of our knowledge, the first work towards *open* domain information extraction was DIPRE [20], which leveraged the basic idea of *syntactic bootstrapping*. Similar ideas were adopted by several later work (e.g., [21], [22], [23]). However, these systems usually require substantial human effort to provide manually tagged *seed* instances for every target relation. KnowItAll [1] partially alleviated this practical challenge by specifying *seed* patterns (or rules) instead of seed instances. But this raised a new question about the availability of hand-crafted linguistic patterns. TextRunner [2] later addressed this issue by using a self-supervised learner to

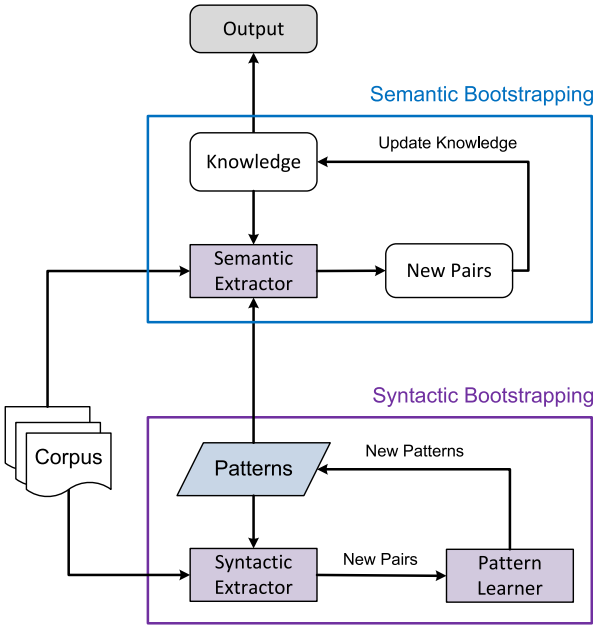


Fig. 5. Combine syntactic and semantic bootstrapping.

automatically identify extraction patterns. Recent work [24] further improved TextRunner by enforcing certain syntactic constraints to suppress incoherent and uninformative extractions. Nonetheless, the precision and scale of the extracted relations remains an issue. Another prominent recent work was the NELL project [3], which viewed information extraction as a multi-task learning problem and employed semi-supervised learning methods. However, NELL still uses syntactic bootstrapping and requires *seed* instances for the learners to bootstrap with. As a result, it suffers similar problems.

Unlike this line of work on syntactic bootstrapping, Probase [4] applied the idea of *semantic bootstrapping* to *isA* relation extraction and showed promising performance. Nonetheless, it only gave empirical results with no performance guarantee. This paper serves as a companion of [4]. We have conducted a detailed theoretical analysis that further explains the working mechanism underlying semantic bootstrapping. So far, we are not aware of similar study on any previous *isA* relation extraction algorithm. It is our hope that our work in this paper could inspire future research on formal analysis of information extraction systems.

On the other hand, semantic bootstrapping does not address the issue of how to obtain high-quality syntactic patterns. There has also been a lot of work on how to obtain good syntactic patterns. For example, Riloff and Jones [23] proposed a *multi-level bootstrapping* architecture, by introducing a *meta-bootstrapping* stage before the standard syntactic bootstrapping framework. Patwardhan and Riloff [25] built a classifier that can evaluate the relevance of an extraction pattern to a piece of text in the corpus. Semantic bootstrapping is orthogonal to this line of work. It is interesting future work to combine both to develop a perhaps more powerful system that consists of two layers: the bottom layer is syntactic bootstrapping, whose output is a set of high-quality extraction patterns; the top layer is semantic bootstrapping, whose input is the output from syntactic bootstrapping, and whose output is the high-quality pairs extracted. This architecture is visualized in Fig. 5.

Finally, it is worth mentioning that semantics-based techniques have been studied for decades in natural language processing (NLP) research, as were detailed in the recent survey by Cambria and White [26]. Unlike pure syntax-based approaches which basically adopt a "bag-of-words" model, semantics-based approaches take a "bag-of-concepts" view that focuses on detecting the intrinsic meaning underlying the text. Knowledge representation lies in the center of these semantics-based approaches, with the aim of building universal taxonomies or Web ontologies [26]. Except for the effort on automatically extracting worldly facts from large open-domain corpus that has been elaborated extensively throughout this paper, there are also many other popular Semantic Web projects (e.g., Annotea [27], SIOC [28], and SKOS [29]). Researchers have also been attempting semantics-based techniques that go beyond encoding subsumption knowledge in taxonomies and ontologies. For example, there is a recent move towards *sentic computing* [30], which presents an approach to concept-level sentiment analysis based on graph mining and dimensionality reduction techniques. Incorporating richer knowledge and semantics into semantic bootstrapping will be a very interesting direction for future work.

8 Conclusion

In this paper, we revisited a semantic bootstrapping framework for open-domain *isA* relation extraction that differs from previous work based on syntactic bootstrapping. Rather than seeking more and more syntactic patterns as the iteration proceeds, semantic bootstrapping uses a fixed set of patterns and leverages the knowledge identified in previous rounds to help extract more knowledge. We gave both theoretical and empirical study of its performance. We demonstrate that semantic bootstrapping can indeed achieve very high precision while retaining good recall on large-scale corpus.

Acknowledgments

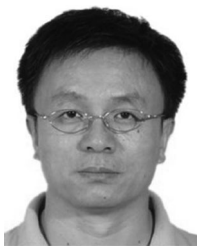
Kenny Q. Zhu is the corresponding author.

References

[1] O. Etzioni, et al., "Web-scale information extraction in knowitall," in *Proc. 13th Int. Conf. World Wide Web*, 2004, pp. 100–110.
[2] M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni, "Open information extraction from the web," in *Proc. 20th Int. Joint Conf. Artif. Intell.*, 2007, pp. 2670–2676.
[3] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. H. Jr., and T. M. Mitchell, "Toward an architecture for never-ending language learning," in *Proc. Conf. Artif. Intell.*, 2010, pp. 1306–1313.
[4] W. Wu, H. Li, H. Wang, and K. Q. Zhu, "Probase: A probabilistic taxonomy for text understanding," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2012, pp. 481–492.
[5] M. A. Hearst, "Automatic acquisition of hyponyms from large text corpora," in *Proc. 14th Conf. Comput. Linguistics*, 1992, pp. 539–545.
[6] R. Snow, D. Jurafsky, and A. Y. Ng, "Learning syntactic patterns for automatic hypernym discovery," in *Proc. 17th Int. Conf. Neural Inf. Process. Syst.*, 2005, pp. 1297–1304.
[7] N. N. Dalvi, A. Machanavajjhala, and B. Pang, "An analysis of structured data on the web," *Proc. VLDB Endowment*, vol. 5, no. 7, pp. 680–691, 2012.
[8] Y. Song, H. Wang, Z. Wang, H. Li, and W. Chen, "Short text conceptualization using a probabilistic knowledgebase," in *Proc. 22nd Int. Joint Conf. Artif. Intell.*, 2011, pp. 2330–2336.

[9] J. Wang, H. Wang, Z. Wang, and K. Q. Zhu, "Understanding tables on the web," in *Proc. 31st Int. Conf. Conceptual Modeling*, 2012, pp. 141–155.
[10] Y. Wang, H. Li, H. Wang, and K. Q. Zhu, "Concept-based web search," in *Proc. 31st Int. Conf. ER Conceptual Modeling*, 2012, pp. 449–462.
[11] J. Bunge and M. Fitzpatrick, "Estimating the number of species," *J. Amer. Statistical Assoc.*, vol. 88, no. 421, pp. 364–373, 1993.
[12] M. Charikar, S. Chaudhuri, R. Motwani, and V. R. Narasayya, "Towards estimation error guarantees for distinct values," in *Proc. 19th ACM SIGMOD-SIGACT-SIGART Symp. Principles Database Syst.*, 2000, pp. 268–279.
[13] S. P. Ponzetto and M. Strube, "Deriving a large-scale taxonomy from wikipedia," in *Proc. 22nd Nat. Conf. Artif. Intell.*, 2007, pp. 1440–1445.
[14] S. Blohm, P. Cimiano, and E. Stemle, "Harvesting relations from the web-quantifiying the impact of filtering functions," in *Proc. 22nd Nat. Conf. Artif. Intell.*, 2007, pp. 1316–1321.
[15] C. Fellbaum, Ed., *WordNet: An Electronic Lexical Database*. Cambridge, MA, USA: MIT Press, 1998.
[16] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: A collaboratively created graph database for structuring human knowledge," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2008, pp. 1247–1250.
[17] D. Freitag and A. McCallum, "Information extraction with HMM structures learned by stochastic optimization," in *Proc. 17th Nat. Conf. Artif. Intell. 12th Conf. Innovative Appl. Artif. Intell.*, 2000, pp. 584–589.
[18] S. Soderland, "Learning information extraction rules for semi-structured and free text," *Mach. Learn.*, vol. 34, no. 1–3, pp. 233–272, 1999.
[19] A. McCallum, "Efficiently inducing features of conditional random fields," in *Proc. 19th Conf. Uncertainty Artif. Intell.*, 2003, pp. 403–410.
[20] S. Brin, "Extracting patterns and relations from the world wide web," in *Proc. Int. Workshop World Wide Web Databases*, 1998, pp. 172–183.
[21] E. Agichtein and L. Gravano, "Snowball: Extracting relations from large plain-text collections," in *Proc. 5th ACM Conf. Digit. Libraries*, 2000, pp. 85–94.
[22] M. Pasca, D. Lin, J. Bigham, A. Lifchits, and A. Jain, "Organizing and searching the world wide web of facts-step one: The one-million fact extraction challenge," in *Proc. 21st Nat. Conf. Artif. Intell.*, 2006, pp. 1400–1405.
[23] E. Riloff and R. Jones, "Learning dictionaries for information extraction by multi-level bootstrapping," in *Proc. 16th Nat. Conf. Artif. Intell. 11th Innovative Appl. Artif. Intell. Conf. Innovative Appl. Artif. Intell.*, 1999, pp. 474–479.
[24] A. Fader, S. Soderland, and O. Etzioni, "Identifying relations for open information extraction," in *Proc. Conf. Empirical Methods Natural Language Process.*, 2011, pp. 1535–1545.
[25] S. Patwardhan and E. Riloff, "Effective information extraction with semantic affinity patterns and relevant regions," in *Proc. Joint Conf. Empirical Methods Natural Language Process. Comput. Natural Language Learn.*, 2007, pp. 717–727.
[26] E. Cambria and B. White, "Jumping NLP curves: A review of natural language processing research [review article]," *IEEE Comp. Int. Mag.*, vol. 9, no. 2, pp. 48–57, May 2014.
[27] J. Kahan and M. Koivunen, "Annotea: An open RDF infrastructure for shared web annotations," in *Proc. 10th Int. Conf. World Wide Web*, 2001, pp. 623–632.
[28] J. G. Breslin, A. Harth, U. Bojars, and S. Decker, "Towards semantically-interlinked online communities," in *Proc. 2nd Eur. Conf. Semantic Web: Res. Appl.*, 2005, pp. 500–514.
[29] T. Baker, S. Bechhofer, A. Isaac, A. Miles, G. Schreiber, and E. Summers, "Key choices in the design of simple knowledge organization system (SKOS)," *J. Web Semantics*, vol. 20, pp. 35–49, 2013.
[30] E. Cambria and A. Hussain, *Sentic Computing: Techniques, Tools, and Applications*, vol. 2. Berlin, Germany: Springer, 2012.

Wentao Wu received the BS and MS degrees in computer science from Fudan University, in 2007 and 2010, respectively, and the PhD degree from the University of Wisconsin-Madison, in 2015. He is currently a researcher with the Data Management, Exploration, and Mining Group, Microsoft Research, Redmond, WA. His research interest includes database management systems, distributed systems, big data analytics, data mining, etc.



Hongsong Li received the PhD degree in computer science from Beijing Jiaotong University. He joined Microsoft Research Asia as an associate researcher in 2006. He has been with the Data intelligence and Tool Group, the Web Search and Mining Group, and the Data Management, Analytics and Services Group. His research interests include knowledge base, web data mining, machine learning, and natural language processing.

Haixun Wang received the BS and MS degrees in computer science from Shanghai Jiao Tong University, in 1994 and 1996, respectively, and the PhD degree in computer science from the University of California, in 2000. He joined Google Research, Mountain View, CA, in 2013. From 2009 to 2013, he was with Microsoft Research, Asia, where he led the database team. From 2000 to 2009, he was with IBM Research, Watson, New York. His research interests include text analytics, semantic network and knowledge base, etc.

Kenny Q. Zhu received the BEng degree in electrical engineering and the PhD degree in computer science from the National University of Singapore, in 1999 and 2005, respectively. He is a professor of computer science with Shanghai Jiao Tong University. From 2007 to 2009, he was a postdoctoral researcher with Princeton University. His research interests include text mining, natural language understanding, and knowledge discovery, and are partially supported by the NSFC and Google Faculty Award.