Semantic Bootstrapping: A Theoretical Perspective

(Extended Abstract)

Wentao Wu [§1] Hongsong Li [†2] Haixun Wang [⋄3] Kenny Q. Zhu [‡4]

[§]*Microsoft Research, Redmond, WA, USA*
[1]wentwu@microsoft.com
[†]*Alibaba Group, Hangzhou, Zhejiang, China*
[2]hongsong.lhs@alibaba-inc.com
[⋄]*Facebook, Inc., Menlo Park, CA, USA*
[3]haixun@fb.com
[‡]*Shanghai Jiao Tong University, Shanghai, China*
[4]kzhu@cs.sjtu.edu.cn

***Abstract*—Knowledge acquisition is an iterative process. Most prior work used syntactic bootstrapping approaches, while semantic bootstrapping was proposed recently. Unlike syntactic bootstrapping, 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 on extracting *isA* relation. Nonetheless, the working mechanism of semantic bootstrapping remains elusive. In this extended abstract, we present a theoretical analysis as well as an experimental study to provide deeper insights into semantic bootstrapping.**

I. Introduction

The problem of extracting *isA* relations in the *open* domain has been studied for years. Most existing systems, such as KnowItAll [1] and TextRunner [2], 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. We refer to this idea as *syntactic bootstrapping*.

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 [3]. Unlike that, Wu et al. [3] 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 [4]) and relies on using existing knowledge (e.g., the pairs already extracted with their frequency) to understand more text and acquire more knowledge (Section II). This approach demonstrates exceptional strength in disambiguating otherwise unaccessible pairs and thus achieves superb precision while maintaining good recall in the extracted pairs.

Nonetheless, the underlying working mechanism of semantic bootstrapping remains elusive in [3]: were the results reported just by chance? In this extended abstract, 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 iteration), which are usually of high quality in practice. Our experimental evaluation results substantiate the theoretical analysis.

II. Semantic Bootstrapping

isA relation can be extracted from sentences that match any of the Hearst patterns, e.g., "... in countries **such as** *China*, *Japan*, ..." 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 $X_s = \{x_1, ..., x_m\}$ 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 $Y_s = \{y_1, ..., y_n\}$ is the set of all candidate sub-concepts.

The 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 [3]:

(P1) For most sentences, only one $x \in X_s$ is valid.
(P2) The closer a $y \in Y_s$ is to $\langle P\rangle$, the more likely y is valid.
(P3) If $y_k \in Y_s$ is valid, then y_1, ..., y_{k-1} are all valid.

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 i-th 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.

By (P1), in the case of $|X_s| > 1$, we need to decide the valid 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.

Assume that we have identified the super-concept $X_s = \{x\}$. The next task is to find the sub-concepts from Y_s. Based on (P2) and (P3), the strategy is to find the largest k such that the likelihood $p(y_k|x)$ is above a threshold.

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

$\Gamma \leftarrow \emptyset$;
$i \leftarrow 1$;
while $true$ **do**
  $\Delta_i \leftarrow \emptyset$;
  foreach $s \in S$ **do**
    $X_s, Y_s \leftarrow ExtractCandidates(s)$;
    if $|X_s| > 1$ **then**
      $X_s \leftarrow DetectSuper(X_s, Y_s, \Gamma_{i-1})$;
    if $|X_s| = 1$ **then**
      $Y_s \leftarrow DetectSub(X_s, Y_s, \Gamma_{i-1})$;
      add valid pairs to Δ_i;
    end
  end
  break if $\Delta_i = \emptyset$;
  $\Gamma_i \leftarrow \Gamma_{i-1} \cup \Delta_i$;
  $i \leftarrow i + 1$;
end
return Γ;

III. Summary of Theoretical Analysis

We first 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 extract new valid pairs from a sentence into Γ, Algorithm 1 is guaranteed to terminate. The efficiency depends on the number of iterations it executes.

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

Here $q = 1-\theta$, where θ ($0 < \theta < 1$) is probability that a sub-concept y in Δ_i serves as the "boundary" y_k specified in (P3). Therefore, $0 < q < 1$. As a result, $\gamma < 1$ and Algorithm 1 ends in $O(\log |\Gamma|)$ iterations. In practice we expect $\gamma \approx \frac{1}{2}$ for most iterations except for the last few ones.

We next 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 by the algorithm.

Theorem 2. *Let* P_1 *and* P_2 *be the precision of* Δ_1 *and* Δ_2. *The precision* P *of* Γ *is* $P = \frac{\alpha P_1 + 2P_2}{\alpha + 2 - \beta}$, *where* $\alpha = \frac{|\Delta_1|}{|\Delta_2|}$ *and* $\beta = \frac{|S|}{|\Delta_2|}$. *Since* $\alpha \geq 0$ *and* $\beta \geq 0$, *we have* $P \geq \frac{2}{2+\alpha} P_2$.

Theorem 2 suggests a lower bound of P that only depends on α and P_2. Since $0 \leq \alpha \leq 1$, we then have $P \geq \frac{2}{3} P_2$, regardless of which α we have. 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 could expect that the lower bound is very close to P_2.

IV. 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 by Wu et al. [3].

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

Round i	$\lvert\Delta_i\rvert$	$\lvert\Delta_i\rvert^u$	$\lvert\Omega_i\rvert$
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

TABLE I. The number of *isA* pairs extracted

We observe from Table I 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 mentioned in Section III.

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

TABLE II. Precision of the pairs extracted

In Table II, we further examined the precision of the pairs (P_i) and the overall precision (Q_i), for each round i of iteration. We find that the overall precision matches our lower bound developed in Section III quite well. According to Theorem 2, the overall precision $P \geq \frac{2}{2+\alpha} P_2$. According to Table II, 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., $Q_{11} = 0.9395$ as in Table II).

References

[1] O. Etzioni, M. J. Cafarella, D. Downey, S. Kok, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates, "Web-scale information extraction in knowitall," in *WWW*, 2004, pp. 100–110.

[2] M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni, "Open information extraction from the web," in *IJCAI*, 2007.

[3] W. Wu, H. Li, H. Wang, and K. Q. Zhu, "Probase: a probabilistic taxonomy for text understanding," in *SIGMOD*, 2012.

[4] M. A. Hearst, "Automatic acquisition of hyponyms from large text corpora," in *COLING*, 1992, pp. 539–545.