Probase: A Probabilistic Taxonomy for Text Understanding

Wentao Wu [1]* Hongsong Li [2] Haixun Wang [2] Kenny Q. Zhu [3]*

[1]University of Wisconsin, Madison, WI, USA
[2]Microsoft Research Asia, Beijing, China
[3]Shanghai Jiao Tong University, Shanghai, China
wentaowu@cs.wisc.edu, {hongsli,haixunw}@microsoft.com, kzhu@cs.sjtu.edu.cn

ABSTRACT

Knowledge is indispensable to understanding. The ongoing information explosion highlights the need to enable machines to better understand electronic text in human language. Much work has been devoted to creating universal ontologies or taxonomies for this purpose. However, none of the existing ontologies has the needed depth and breadth for "universal understanding". In this paper, we present a universal, probabilistic taxonomy that is more comprehensive than any existing ones. It contains 2.7 million concepts harnessed automatically from a corpus of 1.68 billion web pages. Unlike traditional taxonomies that treat knowledge as black and white, it uses probabilities to model inconsistent, ambiguous and uncertain information it contains. We present details of how the taxonomy is constructed, its probabilistic modeling, and its potential applications in text understanding.

Categories and Subject Descriptors

H.3.4 [**Information Storage and Retrieval**]: Systems and Software

General Terms

Algorithms, Design, Experimentation

Keywords

Knowledgebase, Taxonomy, Text Understanding

1. INTRODUCTION

The Web has become one of the largest data repository in the world. But data on the Web is mostly text in natural languages, which is poorly structured and difficult for machines to access. To unlock the trove of information, we must enable machines to process web data automatically. In other words, machines need to understand text in natural languages.

An important question is, what does the word "*understand*" mean here? Consider the following example. For human beings, when we see "25 Oct 1881", we recognize it as a date, although most of us do not know what it is about. However, if we are given a little more context, say the date is embedded in the following piece of short text "Pablo Picasso, 25 Oct 1881, Spain", most of us would have guessed (correctly) that the date represents Pablo Picasso's birthday. We are able to do this because we possess certain knowledge, and in this case, "*one of the most important dates associated with a person is his birthday*". As another example, consider the following two sentences that contain the "*such as*" phrase: "*household pets* other than *dogs* such as *cats* ..." and "*household pets* other than *animals* such as *reptiles* ... ". Humans do not feel that they are ambiguous. Subconsciously they parse the sentences in different ways to obtain the correct semantics: "household pets such as cats" for the first sentence, and "animals such as reptiles" for the second, while machines do not understand why "dogs such as cats" and "household pets such as reptiles" are improbable interpretations. The reason is because humans have background knowledge.

It turns out that what takes a human to understand the above two examples is nothing more than the knowledge about *concepts* (e.g., persons, animals, etc.) and the ability to *conceptualize* (e.g., cats are animals). This is no coincidence. Psychologist Gregory Murphy began his highly acclaimed book with the statement "*Concepts are the glue that holds our mental world together*" [22]. Nature magazine book review pointed out "*Without concepts, there would be no mental world in the first place*" [4].

People use taxonomies and ontologies to represent and organize concepts. Understanding text in the open domain (e.g., understanding text on the Web) is very challenging. The diversity and complexity of human language requires the taxonomy / ontology to capture concepts with various granularities in *every* domain. Although many taxonomies / ontologies exist in specific domains, only a handful of general-purpose ones (Table 1) are available. These taxonomies / ontologies share two key limitations, which make them less effective in general-purpose understanding.

First, existing taxonomies have *limited concept space.* Most taxonomies are constructed by a manual process known as *curation*. This laborious, time consuming, and costly process limits the scope and the scale of the taxonomies thus built. For example, the Cyc project [18], after 25 years of continuing effort by many domain knowledge experts, contains about 120,000 concepts. To overcome this bottleneck, some open domain knowledgebases, e.g., Freebase [5], rely on community efforts to increase the scale. However, while they have near-complete coverage of several specific concepts (e.g., *books*, *music* and *movies*), they lack general coverage of many other concepts. More recently, automatic taxonomy construction approaches, such as KnowItAll [12], TextRunner [2], YAGO [35], and NELL [7], have been in focus, but they still have a limited scale and coverage in terms of concept space.

Limited concept space restricts understanding at *coarse* levels. Consider the following sentence:

*This work was done at Microsoft Research Asia. Kenny Q. Zhu was partially supported by NSFC Grants 61033002 and 61100050.

SIGMOD'12, May 20–24, 2012, Scottsdale, Arizona, USA.

EXAMPLE 1. *"How do we compete with the largest companies in China, India, Brazil, etc.?"*

What countries are China, India, and Brazil? What are the largest companies there? Most of the existing taxonomies contain the concept *company*, and have China, India, and Brazil in the concept of *country*. However, these concepts are too general and do not help understanding. To uncover the semantics encoded here, machines need knowledge of much *finer* concepts such as *largest companies in China*, *developing countries*, and *BRIC countries*. Unfortunately, none of the existing taxonomies contains these concepts.

Fine level of understanding is also desirable in many other important tasks, such as *named entity recognition* (NER) [23] and *word sense disambiguation* (WSD) [24]. NER seeks to locate and tag real-world entities mentioned in the text with their *types* (i.e., concepts). While early research on NER is confined to *coarse-grained* named entity classes such as *person* and *location*, it is generally agreed that *fine-grained* NER [14, 15] (i.e., by using more specific subcategories) is more beneficial for a wide range of web applications, including Information Retrieval (IR), Information Extraction (IE), or Query-Answering (QA). WSD governs the process of identifying the sense (i.e., meaning) of a word used in the text. Knowledge sources such as taxonomies and ontologies are fundamental components of WSD, which distinguishes senses of words by their categories (i.e., concepts). It has been widely observed that different NLP applications require different sense granularities in order to best exploit word sense distinctions [33]. All these applications require the taxonomies / ontologies to contain a *rich* set of concepts, with various granularities.

Existing Taxonomies	Number of Concepts
Freebase [5]	1,450
WordNet [13]	25,229
WikiTaxonomy [26]	111,654
YAGO [35]	352,297
DBPedia [1]	259
ResearchCyc [18]	$\approx$ 120,000
KnowItAll [12]	N/A
TextRunner [2]	N/A
OMCS [31]	N/A
NELL [7]	123
Probase	**2,653,872**

Table 1: Scale of open-domain taxonomies

Second, existing taxonomies treat knowledge as *black and white*. They believe that a knowledgebase should provide standard, well-defined and consistent reusable information, and therefore all concepts and relations included in these taxonomies are kept "religiously" clean. However, many real-world concepts, such as *large company*, *best university* and *beautiful city* do not have a *definite* boundary, and are intrinsically *vague*. Many of these vague concepts are useful in fine-grained understanding. For instance, there are tens of thousands of universities in the world, while only hundreds of them are considered the *best*. Moreover, automatic taxonomy construction processes, while reducing costs and improving productivity, are never perfect. They can introduce errors and inconsistencies into the taxonomies thus built. Without a good way to model the vagueness and inconsistencies, all existing taxonomies choose to either exclude the vague concepts or ignore the inconsistencies.

In this paper, we argue that accommodating and modeling such uncertainties in a taxonomy can be very useful in conceptualization. To see this, let us take a deeper look at the sentence in Example 1. Depending on the application, we may need to understand: i) what does "*largest companies*" mean? and ii) what does "*China, India, Brazil*" signify?

There are obviously millions of companies in these countries, but only a handful of them are considered *largest companies*. Since the term "*largest*" is *subjective*, to understand what are *largest companies*, we concretize this vague concept to a set of its most *typical* instances, such as China Mobile, Tata Group, and PetroBras. On the other hand, each of the three instances *China*, *India*, and *Brazil*, may be interpreted as *country*, *big country*, *developing country*, *BRIC country*, or *emerging market*. All of those choices are correct, but as a group together, *BRIC country* and *emerging market* might be the best abstractions, because they are the most *typical* and the "tightest" concept to characterize the three instances. With this generalization, one can even suggest a fourth instance, *Russia*, to complete the sentence.

From the above example, we can see that conceptualization can proceed in two directions:

- *Instantiation*: given a concept, inferring its typical and likely instances (e.g., from *largest company* to *China Mobile*, *Tata Group*, etc).

- *Abstraction*: given one or multiple instances[1], inferring the typical and likely concepts they belong to (e.g., from *China*, *India*, *Brazil* to *emerging market* or *BRIC country*);

In either direction, uncertainties are inherent, and probabilities play an important role in the inference.

In this paper, we introduce Probase, a universal, general-purpose, probabilistic taxonomy automatically constructed from a corpus of 1.6 billion web pages. The Probase taxonomy is unique in three aspects:

1. It is built by a novel framework, which consists of an iterative learning algorithm to extract *isA* pairs from web text (see Section 2), and a taxonomy construction algorithm to connect these pairs into a hierarchical structure (see Section 3). The resulting taxonomy has the highest precision (92.8%) and largest scale reported so far in automated web-scale taxonomy inference research.

2. It is the first general-purpose taxonomy that takes a probabilistic approach to model the knowledge it possesses. Knowledge in Probase is no longer black and white. Each fact or relation is associated with some probabilities to measure its plausibility and typicality. Plausibility is useful for detecting errors and integrating heterogeneous knowledge sources, while typicality is useful for conceptualization and inference. Such a probabilistic treatment allows Probase to better capture the semantics of human languages (see Section 4). Recent work [34, 39, 37] based on this probabilistic treatment of knowledge in Probase demonstrates the effectiveness of this framework, which will be discussed further in Section 5.3.

3. It is the largest general-purpose taxonomy fully automatically constructed from HTML text on the web. Probase has a huge concept space with almost 2.7 million concepts, 8 times larger than that of YAGO, which makes it the largest taxonomy in terms of concept space (Table 1). Besides popular concepts such as "cities" and "musicians", which are already in almost every general-purpose taxonomy, it also has tens of thousands of specific concepts such as "renewable energy technologies", "meteorological phenomena" and "common

[1]Probase also supports abstraction from a mixture of instances, attributes, and actions. For example, inferring from *headquarter, apple* to *company*, or from *Germany invaded Poland* to *war*. However, in this paper, we focus on concepts and instances.

sleep disorders", which cannot be found in Freebase, Cyc, or any other taxonomies (see Section 5).

More information about Probase , including Probase-enabled applications [34, 39, 37], and a small excerpt of the Probase taxonomy, can be found at `http://research.microsoft.com/probase/`.We are currently working to make the Probase taxonomy available to the public.

2. ITERATIVE EXTRACTION

We present a novel iterative learning framework that aims at acquiring knowledge with high precision and high recall. Knowledge acquisition consists of two phases: i) information extraction, and ii) data cleansing and integration. A lot of work has been done in data cleansing and integration [17, 19, 20] for Probase. In this paper, we focus on the first phase: information extraction.

Information extraction is an iterative process. Most existing approaches bootstrap on syntactic patterns, that is, each iteration finds more syntactic patterns for subsequent extraction. Our approach, on the other hand, bootstraps directly on knowledge, that is, we use existing knowledge to understand the text and acquire more knowledge. In the following, we describe the limitations of state-of-the-art work in Section 2.1, existing problems and challenges in Section 2.2, and our new iterative learning framework in Section 2.3.

2.1 Syntactic vs. Semantic Iteration

State-of-the-art information extraction methods, including KnowItAll [12], TextRunner [2], and NELL [7], rely on an iterative (bootstrapping) approach. It starts with a set of seed examples and/or seed patterns. From the examples, it derives new patterns that fit the examples. Then, it uses the new patterns to extract more examples from the data. The iterative process is at the syntax level. It has limitations which prevent deep knowledge acquisition. Our goal is to break this barrier and perform extraction at semantic, or knowledge level.

Syntactic Iteration. High quality syntactic patterns are valuable to information extraction. Assume we are interested in finding *isA* relationships (the most important relationship in knowledge bases). We can start with the Hearst patterns [16].

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*

Table 2: The Hearst patterns (*NP* stands for *noun phrase*)

Using the above Hearst patterns, we can derive knowledge from text. For example, given a sentence, "... domestic animals such as cats ...", we obtain the relationship: "cat *isA* animal". The idea of syntactic iteration is that, in order to find more relationships, we need more syntactic patterns. Thus, we endeavor to discover more syntactic patterns that exist among the current *isA* pairs (which include "cat *isA* animal") and use them to obtain more *isA* pairs.

This process has been adopted by most information extraction approaches. However, it focuses on syntax only, and has these limitations:

- *Syntactic patterns have limited extraction power.* Natural languages are ambiguous, and syntactic patterns alone are not powerful enough to deal with such ambiguity. Consider the sentence "... animals other than dogs such as cats ...". KnowItAll will extract (cat *isA* dog) rather than (cat *isA* animal). Because the syntactic structure is inherently ambiguous (as we described in Section 1), refining the syntactic rules will not help in such a case.

- *High quality syntactic patterns are rare.* The syntactic patterns obtained from bootstrapping often have low quality. For instance, assume we want to find instances of countries, that is (x *isA* country). From a set of seed countries, we may obtain syntactic patterns such as "war with x", "invasion of x", and "occupation of x". But, from such patterns, we might derive wrong instances, for example, "x = planet Earth." This problem is known as semantic drift [7]. To deal with the problem, sophisticated "discriminators" are created to remove syntactic patterns of low quality. Unfortunately, for *isA* relationships, the remaining patterns are mostly Hearst patterns. Thus the central idea of "more syntactic patterns can produce more results" is not really valid.

- *Recall is sacrificed for precision.* Because natural languages are ambiguous, and syntactic patterns have low quality, state-of-the-art approaches have to sacrifice recall for precision. For instance, when extracting *isA* pairs, they focus on instances that are *proper nouns*. It thus cannot derive the knowledge (cat *isA* animal) from simple sentences like "animals such as cats". But such *isA* relationships are essential in creating knowledge taxonomies. Furthermore, most approaches also restrict the concept to be a noun instead of a noun phrase. For example, from a sentence "... industrialized countries such as US and Germany ...", only (US *isA* country) is extracted, instead of (US *isA* industrialized country).

Semantic Iteration. Probase performs iterative learning at the *knowledge* or semantic level. Given the sentence, "domestic animals other than dogs such as cats", Probase realizes that there are two possible readings: (cat *isA* dog) and (cat *isA* domestic animal). If Probase knows nothing about cats, dogs, and domestic animals (which is the case in the first iteration), it cannot decide which one is more probable. Thus, the sentence is discarded. However, when the second iteration begins, Probase already has acquired a lot of knowledge, and it knows that (domestic animal *isA* animal), and the frequency of (cat *isA* animal) is much higher than (cat *isA* dog). When this difference is above a certain threshold, it can correctly choose between the two possible readings.

In other words, in Probase, the power of obtaining new knowledge does not come from the use of more syntactic patterns. In fact, a fixed set of syntactic patterns, i.e., Hearst patterns, are used in each iteration. Rather, the power comes from the existing knowledge.

Some previous work in named entity recognition (NER) has also addressed similar issues. Downey et al. [10] brought up the following problem: how do we know if "... companies such as Proctor and Gamble ..." is talking about two companies {*Proctor*, *Gamble*} or a single company whose name is *Proctor and Gamble*? Their idea is to use Pointwise Mutual Information (PMI) to measure the association between *Proctor* and *Gamble*. This is similar to our approach discussed in Section 2.3.3, where we note that the frequency of *Proctor and Gamble* as one term is much higher than the frequency of *Proctor* appearing alone. Compared with their work, our approach is more general since we are not limited to PMI. In fact, we can take advantage of any existing knowledge we have already learned. For example, if we know we are talking about "cartoons" (the super-concept, which is the context of an *isA* extraction) , it

is more likely that *Tom and Jerry* should be treated as a single instance rather than two instances {*Tom*, *Jerry*}.

2.2 Problem Definition

In this paper, we present our iterative learning framework in the setting of extracting *isA* pairs from web documents. Unlike state-of-the-art information extraction methods that rely on discovering additional syntactic patterns for obtaining new knowledge, our iteration uses a fixed set of syntactic patterns (Hearst patterns), and relies on using existing knowledge to understand more text, and acquire more knowledge.

From a sentence that matches any of the Hearst patterns, we want to obtain

$$s = \{(x, y_1), (x, y_2), ..., (x, y_m)\}$$

where x is the superordinate concept (or super-concept), and $\{y_1, \cdots, y_m\}$ are its subordinate concept (or sub-concept). For example, from the sentence

"... in tropical countries **such as** *Singapore*, *Malaysia*, ..."[2]

we derive s ={(tropical country, Singapore), (tropical country, Malaysia)}.

Natural languages are rife with ambiguities, and syntactic patterns alone cannot deal with the ambiguity. Here are some examples (found in our corpus):

EXAMPLE 2. *Example sentences.*

1) ... animals other than dogs **such as** *cats* ...

2) ... classic movies **such as** *Gone with the Wind* ...

3) ... companies **such as** *IBM, Nokia, Proctor and Gamble* ...

4) ... representatives in North America, Europe, the Middle East, *Australia, Mexico, Brazil, Japan, China,* **and other** countries ...

If we rely on syntactic patterns alone, 1) *dogs* will be extracted as the super-concept, not *animals*; 2) nothing is extracted since *Gone with the Wind* is not a noun phrase; 3) *Proctor* and *Gamble* are treated as two companies; 4) *North America*, *Europe*, and *the Middle East* are mistakenly extracted as countries.

2.3 The Framework

Our framework focuses on *understanding*. In many cases, semantics is required to supplement the syntax for correct extraction. As the iterations progress, we acquire more and more knowledge. Using the knowledge, we can have a better understanding of the semantics, which adds to the power of our extraction framework.

Specifically, we propose an iterative learning process. In each round of information extraction, we accumulate knowledge for which we have high confidence to be correct. We then use this knowledge in the next round to help us extract information we missed previously. We perform this process *iteratively* until no more information can be extracted.

More specifically, let Γ denote the knowledge we currently have, i.e., the set of *isA* pairs that we have discovered. For each $(x, y) \in \Gamma$, we also keep a count $n(x, y)$, which indicates how many times (x, y) is discovered. Initially, Γ is empty. We search for *isA* pairs in the text, and we use Γ to help identify valid ones among them. We expand Γ by adding the newly discovered pairs, which further enhances our power to identify more valid pairs. Table 3 summarizes important notations used throughout this paper.

[2]The underlined term is the super-concept, and the italicized terms are its sub-concepts.

Notation	Meaning
Γ	the set of *isA* pairs extracted from the corpus
(x, y)	an *isA* pair with super-concept x and sub-concept y
$n(x, y)$	# of times (x, y) is discovered in the corpus
s	a sentence that matches any of the Hearst patterns
X_s	candidate super-concepts of sentence s
Y_s	candidate sub-concepts of sentence s
x^i	a concept x with sense i
T_x^i	a local taxonomy with root x^i
$\mathcal{P}(x, y)$	plausibility of the *isA* pair (x, y)
$\mathcal{T}(i\|x)$	typicality of the instance i given the concept x
$\mathcal{T}(x\|i)$	typicality of the concept x given the instance i

Table 3: Notations

Algorithm 1: *isA* extraction

Input: S, sentences from web corpus that match the Hearst patterns
Output: Γ, set of *isA* pairs

```
Γ ← ∅;
repeat
    foreach s ∈ S do
        X_s, Y_s ← SyntacticExtraction(s) ;
        if |X_s| > 1 then
            X_s ← SuperConceptDetection(X_s, Y_s, Γ);
        end
        if |X_s| = 1 then
            Y_s ← SubConceptDetection(X_s, Y_s, Γ);
            add valid isA pairs to Γ;
        end
    end
until no new pairs added to Γ;
return Γ;
```

Algorithm 1 outlines our method at a high level. It repeatedly scans the set of sentences until no more pairs can be identified. Procedure *SyntacticExtraction* finds *candidate* super-concepts X_s and *candidate* sub-concepts Y_s from a sentence s. If more than one candidate super-concepts exist, we call procedure *SuperConceptDetection* to reduce X_s to a single element. Then, procedure *SubConceptDetection* filters out unlikely sub-concepts in Y_s. Finally, we add newly found *isA* pairs to the result. Due to the new results, we may be able to identify more pairs, so we scan the sentences again. We describe the three procedures in detail below.

2.3.1 Syntactic Extraction

Procedure *SyntacticExtraction* detects candidate super-concepts X_s and sub-concepts Y_s in a sentence s.

As shown in sentence 1) of Example 2, the noun phrase that is *closest* to the pattern keywords may *not* be the correct super-concept. Therefore, X_s should contain all possible noun phrases. For sentence 1), we identify candidate super-concepts as $X_{1)} = \{animals, dogs\}$. As in some previous work [12, 27], we further require that every element in X_s must be a noun phrase in *plural form*. As a result, for the sentence "... countries other than Japan **such as** *USA* ...", the set of candidate super-concepts contains "countries" but not "Japan".

It is more challenging to identify Y_s. First, as shown in sentence 2), sub-concepts may *not* be noun phrases. Second, as shown in sentence 3), delimiters such as "*and*" and "*or*" may themselves appear in valid sub-concepts. Third, as shown in sentence 4), it is often difficult to detect where the list of sub-concepts begins or ends. Therefore, we adopt a rather conservative approach at this stage by including all potential sub-concepts into Y_s.

Based on the Hearst pattern in use, we first extract a list of candidates by using ',' as the delimiter. For the *last* element, we also use

"and" and "or" to break it down. Since words such as "and" and "or" may or may not be a delimiter, we keep all possible candidates in Y_s. For instance, given sentence 3) in Example 2, we have $Y_{3)} = \{IBM, Nokia, Proctor, Gamble, Proctor\ and\ Gamble\}$.

2.3.2 Super-Concept Detection

In case $|X_s| > 1$, we must remove unlikely super-concepts from X_s until only one super-concept remains. We use a probabilistic approach for super-concept detection.

Let $X_s = \{x_1, \cdots, x_m\}$. We compute likelihood $p(x_k|Y_s)$ for $x_k \in X_s$. Without loss of generality, we assume x_1 and x_2 have the *largest* likelihoods, and $p(x_1|Y_s) \geq p(x_2|Y_s)$. We compute the ratio of likelihood $r(x_1, x_2)$ as follows and then we pick x_1 if the ratio is above a threshold:

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

Since the list of candidate sub-concepts are well know as *coordinate terms* in the literature, which means they are equally important under the super-concept, we assume sub-concepts in $Y_s = \{y_1, \cdots, y_n\}$ are independent given the super-concept, and have

$$r(x_1, x_2) = \frac{p(x_1)\prod_{i=1}^{n} p(y_i|x_1)}{p(x_2)\prod_{i=1}^{n} p(y_i|x_2)}$$

We compute the above ratio as follows: $p(x_i)$ is the percentage of pairs that have x_i as the super-concept in Γ, and $p(y_j|x_i)$ is the percentage of pairs in Γ that have y_j as the sub-concept given x_i is the super-concept. Certainly, not every (x_i, y_j) appears in Γ, especially in the beginning when Γ is small. This leads to $p(y_j|x_i) = 0$, which makes it impossible for us to calculate the ratio. To avoid this situation, we let $p(y_j|x_i) = \epsilon$ where ϵ is a small positive number, when (x_i, y_j) is not in Γ.

As an example, from sentence 1) in Example 2, we obtain $X_{1)} = \{animals, dogs\}$ by syntactic extraction. Intuitively, the likelihood $p(animals|cats)$ should be much higher than $p(dogs|cats)$ in a large corpus, since it is very unlikely for sentences like "... dogs **such as** *cats* ..." to exist, while sentences like "... animals **such as** *cats* ..." are quite common. As a result, the ratio $r(animals, dogs)$ should be large, and "animals" will be picked as the correct super-concept.

However, the above approach cannot find *new* super-concepts. Consider the sentence "... domestic animals other than dogs **such as** *cats* ...". If we have not seen many sentences with "domestic animals" and "cats" together, it is not possible to infer "domestic animals" as a possible super-concept. However, common knowledge tells us that *domestic* animals are also animals. Since Γ has the pair (animals, cats), we can also derive a large likelihood p(*domestic animals*|*cats*), and pick "domestic animals" as the super-concept. In general, for any candidate super-concept x in X_s that is not in Γ, we strip the modifier of x and check the remaining (more general) concept in Γ again. This helps us harvest more specific concepts from the corpus and improve the recall.

2.3.3 Sub-Concept Detection

Assume we have identified the super-concept $X_s = \{x\}$ from a sentence. The next task is to find its sub-concepts from Y_s. In our work, sub-concept detection is based on features extracted from the sentences. Due to lack of space, here we only focus on the most important features, derived from the following two observations.

OBSERVATION 1. *The closer a candidate sub-concept is to the* **pattern keywords***, the more likely it is a valid sub-concept.*

In fact, some extraction methods (e.g., [27]) only take the closest one to improve precision. For example, in sentence 3) of Example 2, *IBM* most likely is a valid sub-concept because it is right after pattern keywords **such as**, and in sentence 4), *China* most likely is a valid sub-concept as it is right before the pattern keywords **and other**.

OBSERVATION 2. *If we are certain a candidate sub-concept at the k-th position from the* **pattern keywords** *is valid, then most likely candidate sub-concepts from position 1 to position $k-1$ are also valid.*

Here, the position of a candidate sub-concept is numbered with respect to its *closeness* to the *pattern keywords*. For instance, in sentence 3) of Example 2, the first sub-concept is *IBM*, the second sub-concept is *Nokia*, and so on, while in sentence 4), the first sub-concept is *China*, the second sub-concept is *Japan*, and so on.

Based on Observation 2, our strategy is to first find the largest scope wherein sub-concepts are all valid, and then address the ambiguity issues inside the scope.

Specifically, we find the largest k such that the likelihood $p(y_k|x)$ is above a threshold, where y_k is the candidate sub-concept at the k-th position from the pattern keywords. If, however, we cannot find any y_k that satisfies the condition, then we assume $k = 1$, provided that y_1 is *well formed* (e.g., it does not contain any delimiters such as "and" or "or"), because based on Observation 1, y_1 is most likely a valid sub-concept.

For example, in sentence 4), we may correctly decide that the list of valid sub-concepts ends at *Australia*, if the likelihood of *Australia* is much larger than the likelihoods of later candidates *the Middle East*, *Europe*, and *North America*.

Then, we study each candidate $y_1, \cdots, y_k$. For any y_i where $1 \leq i \leq k$, if y_i is not ambiguous, we add (x, y_i) to Γ if it is not already there, or otherwise we increase the count $n(x, y_i)$. If y_j is *ambiguous*, that is, we have multiple choices for position j, then we need to decide which one is valid.

Assume we have identified $y_1, \cdots, y_{j-1}$ from position 1 to position $j-1$ as valid sub-concepts, and assume we have two candidates[3] at position j, that is, $y_j \in \{c_1, c_2\}$. We compute the likelihood ratio $r(c_1, c_2)$ as follows and then we pick c_1 over c_2 if the ratio is above a threshold:

$$r(c_1, c_2) = \frac{p(c_1|x, y_1, \cdots, y_{j-1})}{p(c_2|x, y_1, \cdots, y_{j-1})}$$

As before, we assume $y_1, \cdots, y_{j-1}$ are independent given x and y_j, and have:

$$r(c_1, c_2) = \frac{p(c_1|x)\prod_{i=1}^{j-1} p(y_i|c_1, x)}{p(c_2|x)\prod_{i=1}^{j-1} p(y_i|c_2, x)}$$

Here, $p(c_1|x)$ is the percentage of pairs in Γ where c_1 is a sub-concept given x is the super-concept, and $p(y_i|c_1, x)$ is the likelihood that y_i appears as a valid sub-concept in a sentence with x as the super-concept and c_1 as another valid sub-concept.

As an example, consider sentence 3) in Example 2. We have multiple choices for the third candidate, namely *Proctor and Gamble* and *Proctor*. Intuitively, the likelihood for *Proctor and Gamble* is much larger than *Proctor*, since the chance that *Proctor* itself appears as a valid company name is quite low (i.e., it is very unlikely that we can encounter sentences like "... companies **such as** *Proctor* ..."). Therefore, p(*Proctor and Gamble*|*companies*), p(*IBM*|*Proctor and Gamble, companies*), and p(*Nokia*|*Proctor and Gamble, companies*) should all be larger than their counterparts involving *Proctor*. As a result, the ratio r(*Proctor and Gamble*, *Proctor*) should be large, and we thus pick "Proctor and Gamble" as the correct sub-concept.

[3] As before, if we have more than two candidates, we pick the two with the largest likelihoods.

3. TAXONOMY CONSTRUCTION

The previous step produces a large set of *isA* pairs. Each pair represents an edge in the taxonomy. Our goal is to construct a taxonomy from these individual edges.

3.1 Problem Statement

We model the taxonomy as a DAG (directed acyclic graph). A node in the taxonomy is either a *concept* node (e.g., *company*), or an *instance* node (e.g., *Microsoft*). A concept contains a set of instances and possibly a set of sub-concepts. An edge (u, v) connecting two nodes u and v means that u is a super-concept of v. Differentiating concept nodes from instance nodes is natural in our taxonomy: *Nodes without out-edges are instances, while other nodes are concepts*.

The obvious task of creating a graph out of a set of edges is the following: *For any two edges each having a node with the same label, should we consider them as the same node and connect the two edges?* Consider the following two cases:

1. For two edges e_1 = (fruit, apple), e_2 = (companies, apple), should we connect e_1 and e_2 on node "apple"?

2. For two edges e_1 = (plants, tree), e_2 = (plants, steam turbine), should we connect e_1 and e_2 on node "plants"?

The answer to both of the questions is obviously *No*, but how do we decide on these questions?

Clearly, words such as "apple" and "plants" may have multiple meanings (senses). So the challenge of taxonomy construction is to differentiate between these senses, and connect edges on nodes that have the same sense. We further divide the problem into two sub-problems: i) Group concepts by their senses, i.e., decide whether the two *plants* in the 2nd question above mean the same thing; and ii) group instances by their senses, i.e., decide whether the two *apples* in the 1st question mean the same thing. We argue that we only need to solve the first sub-problem, because once we correctly group all the concepts by different senses, we can determine the meaning of an instance by its position in the concept hierarchy, i.e., its meaning depends on all the super-concepts it has.

We attack the problem of taxonomy construction in two steps. First, we identify some properties of the *isA* pairs we have obtained. Second, based on the properties, we introduce two operators that merge nodes belonging to the same sense, and we build a taxonomy using the operators we defined.

3.2 Senses

Let x^i denote a node with label x and sense i. Two nodes x^i and x^j are equivalent iff $i = j$. For an edge (x, y), if (x^i, y^j) holds, then (x^i, y^j) is a possible *interpretation* of (x, y). We denote this as $(x, y) \models (x^i, y^j)$. Given an edge (x, y), there are 3 possible cases for interpreting (x, y):

1. There exists a unique i and a unique j such that $(x, y) \models (x^i, y^j)$. For example, (*planets*, *Earth*). This is the most common case.

2. There exists a unique i and multiple j's such that $(x, y) \models (x^i, y^j)$. For example, (*objects*, *plants*).

3. There exists multiple i's and multiple j's such that $(x, y) \models (x^i, y^j)$. This case is very rare in practice.

Finally, it is impossible that there exist multiple i's but a unique j such that $(x, y) \models (x^i, y^j)$.

3.3 Properties

We reveal some important properties for the *isA* pairs we obtain through the Hearst patterns. In our discussion, we use the following sentences as our running example.

EXAMPLE 3. *A running example.*

a) ... plants **such as** *trees* **and** *grass* ...

b) ... plants **such as** *trees, grass* **and** *herbs* ...

c) ... plants **such as** *steam turbines, pumps,* **and** *boilers* ...

d) ... organisms **such as** *plants, trees, grass* **and** *animals* ...

e) ... things **such as** *plants, trees, grass, pumps,* **and** *boilers* ...

PROPERTY 1. *Let $s = \{(x, y_1), \cdots, (x, y_n)\}$ be the* isA *pairs derived from a sentence. Then, all the x's in s have a unique sense, that is, there exists a unique i such that $(x, y_j) \models (x^i, y_j)$ holds for all $1 \leq j \leq n$.*

Intuitively, it means that sentences like "... plants **such as** *trees* **and** *boilers* ..." are extremely rare. In other words, the super-concept in all the *isA* pairs from a single sentence has the same sense. For example, in sentence a), the senses of the word *plant*s in (*plants*, *trees*) and (*plants*, grass) are the same. Following this property, we denote *isA* pairs from a sentence as $\{(x^i, y_1), \cdots, (x^i, y_n)\}$ by emphasizing that all the x's have the same sense.

PROPERTY 2. *Let $\{(x^i, y_1), \cdots, (x^i, y_m)\}$ denote pairs from one sentence, and $\{(x^j, z_1), \cdots, (x^j, z_n)\}$ from another sentence. If $\{y_1, ..., y_m\}$ and $\{z_1, ..., z_n\}$ are similar, then it is highly likely that x^i and x^j are equivalent, that is, $i = j$.*

Consider sentences a) and b) in Example 3. The set of sub-concepts have a large overlap, so we conclude that the senses of the word *plants* in the two sentences are the same. The same thing cannot be said for sentences b) and c) as no identical sub-concepts are found.

PROPERTY 3. *Let $\{(x^i, y), (x^i, u_1), \cdots, (x^i, u_m)\}$ denote pairs obtained from one sentence, and $\{(y^k, v_1), \cdots, (y^k, v_n)\}$ from another sentence. If $\{u_1, u_2, ..., u_m\}$ and $\{v_1, v_2, ..., v_n\}$ are similar, then it is highly likely that $(x^i, y) \models (x^i, y^k)$.*

This means the word *plants* in sentence d) has the same sense as the word *plants* in sentence a), because their sub-concepts have considerable overlap. This is not true for sentence d) and c). For the same reason, the word *plants* in sentence e) could be interpreted as the sense of *plants* in a) and c) at the same time.

3.4 Node Merging Operations

Based on the three properties, we can immediately develop some mechanisms to join the edges by end-nodes of the same sense.

First, based on Property 1, we know that every super-concept in the *isA* pairs derived from a single sentence has the same sense. Thus, we join such *isA* pairs on the super-concept node (see Figure 1). We call the taxonomy obtained from a single sentence a *local taxonomy*. A local taxonomy with root x^i is denoted as T_x^i.

Second, based on Property 2, given two local taxonomies rooted at x^i and x^j, if the child nodes of the two taxonomies demonstrate considerable similarity, we perform a *horizontal merge* (see Figure 2).

Third, based on Property 3, given two local taxonomies rooted at x^i and y^k, if x^i has a child node y, and the child nodes of the two taxonomies demonstrate considerable similarity, we merge the two

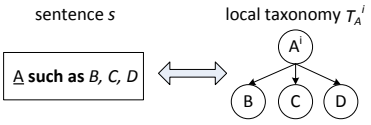


Figure 1: From a single sentence to a local taxonomy

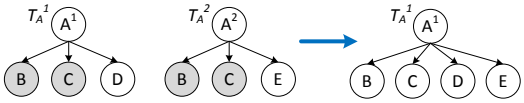
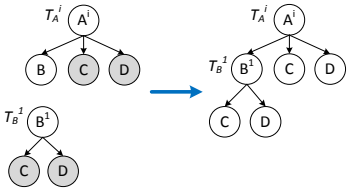


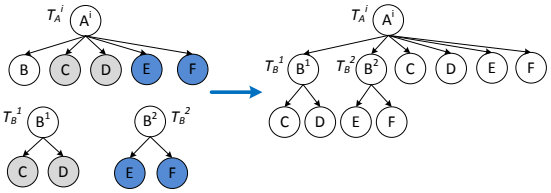
Figure 2: Horizontal merge

taxonomies on the node y. We call this a *vertical merge*, and it is illustrated in Figure 3(a).

A special case for vertical merge is illustrated in Figure 3(b). Both T_B^1 and T_B^2 are vertically mergeable with T_A^i, as the child nodes of T_B^1 and T_B^2 having considerable overlap with the child nodes of T_A^i. However, the child nodes of T_B^1 and T_B^2 do not have considerable overlap, so it is still possible that they represent two senses. The result is two sub-taxonomies as shown in Figure 3(b).



(a) Single sense alignment



(b) Multiple sense alignment

Figure 3: Vertical merge

3.5 Similarity Function

Suppose we use $Child(T)$ to denote the child nodes of a local taxonomy T. Given two local taxonomies T_1 and T_2, a central step lying in both the horizontal and vertical merge operations is to check the overlap of $Child(T_1)$ and $Child(T_2)$. In general, we can define a similarity function $f(A, B)$ such that two sets A and B are *similar* (denoted as $Sim(A, B)$) if $f(A, B) \geq \tau(A, B)$, where $\tau(A, B)$ is some prespecified threshold function.

However, the choice of $f(A, B)$ is not arbitrary. Similarity defined in a relative manner, such as the Jaccard metric, can often lead to unreasonable results. Consider the following example:

Let $A = \{\text{Microsoft, IBM, HP}\}$, $B = \{\text{Microsoft, IBM, Intel}\}$, $C = \{\text{Microsoft, IBM, HP, EMC, Intel, Google, Apple}\}$, and suppose that the super-concept of A, B, and C is "IT company". Let $J(X, Y) = \frac{|X \cap Y|}{|X \cup Y|}$ be the Jaccard similarity of X and Y. Then $J(A, B) = \frac{2}{4} = 0.5$, while $J(A, C) = \frac{3}{7} = 0.43$. If we set the similarity threshold to be 0.5, then A and B are deemed to be similar, while A and C are not. As a result, horizontal merge can be applied to A and B, but not to A and C. This is absurd since A is a *subset* of C.

We thus focus on $f(A, B)$ measured by the *absolute* overlap of A and B. Furthermore, with respect to Property 2 and 3, the more overlapping evidence we have, the more confident we are in performing the merge operations. Therefore, we require $f(A, B)$ to conform to the following property:

PROPERTY 4. *If A, A', B, and B' are any sets s.t. $A \subseteq A'$ and $B \subseteq B'$, then $Sim(A, B) \Rightarrow Sim(A', B')$.*

The simplest way then may be to define $f(A, B) = |A \cap B|$ and let $\tau(A, B)$ equal to some constant δ. Under this setting, we have $f(A', B') \geq f(A, B)$. Hence $f(A, B) \geq \delta \Rightarrow f(A', B') \geq \delta$, namely $Sim(A, B) \Rightarrow Sim(A', B')$.

3.6 The Algorithm

Algorithm 2 summarizes the framework of taxonomy construction. The whole procedure can be divided into three stages, namely, the *local taxonomy construction* stage (line 3-5), the *horizontal grouping* stage (line 6-10), and the *vertical grouping* stage (line 11-19). We first create a local taxonomy from each sentence $s \in S$. We then perform horizontal merges on local taxonomies whose root nodes have the same label. In this stage, small local taxonomies will be merged to form larger ones. Finally, we perform vertical merges on local taxonomies whose root nodes have different labels.

Algorithm 2: Taxonomy construction

Input: S: the set of sentences each containing a number of *isA* pairs.
Output: T: the taxonomy graph.

```
1  Let 𝒯 be the set of local taxonomies;
2  𝒯 ← ∅;
3  foreach s = {(x^i, y_1), ..., (x^i, y_n)} ∈ S do
4  |  Add a local taxonomy T_x^i into 𝒯;
5  end
6  foreach T_x^i ∈ 𝒯, T_x^j ∈ 𝒯 do
7  |  if Sim(Child(T_x^i), Child(T_x^j)) then
8  |  |  HorizontalMerge(T_x^i, T_x^j);
9  |  end
10 end
11 foreach T_x^i ∈ 𝒯 do
12 |  foreach y ∈ Child(T_x^i) do
13 |  |  foreach T_y^m ∈ 𝒯 do
14 |  |  |  if Sim(Child(T_x^i), Child(T_y^m)) then
15 |  |  |  |  VerticalMerge(T_x^i, T_y^m);
16 |  |  |  end
17 |  |  end
18 |  end
19 end
20 Let the graph so connected be T;
21 return T;
```

In Algorithm 2, we perform horizontal grouping before vertical grouping. This order is not necessary. As Theorem 1 dictates, any sequence of merge operations eventually lead to the same taxonomy. However, as Theorem 2 suggests, the total number of merge operations is minimized when horizontal grouping is done before vertical grouping, which is more desirable. The proofs of the two theorems can be found in [40].

THEOREM 1. *Let $\mathcal{T}$ be a set of local taxonomies. Let $\mathbf{O}^\alpha$ and $\mathbf{O}^\beta$ be any two sequences of horizontal and vertical merge operations on $\mathcal{T}$. Assume no further operations can be performed on $\mathcal{T}$ after $\mathbf{O}^\alpha$ or $\mathbf{O}^\beta$. Then, the final graph after performing $\mathbf{O}^\alpha$ and the final graph after performing $\mathbf{O}^\beta$ are identical.*

THEOREM 2. *Let $\mathcal{O}$ be the set of all possible sequences of operations, and let $M = \min\{|\mathbf{O}| : \mathbf{O} \in \mathcal{O}\}$. Suppose $\mathbf{O}^\sigma$ is the sequence that performs all possible horizontal merges first and all possible vertical merges next, then $|\mathbf{O}^\sigma| = M$.*

3.7 Related Work

Automatic taxonomy construction has been extensively studied in the literature [6, 28, 8, 21, 32, 35, 26]. Early work [6, 28, 8] focused on inducing closed-domain taxonomies, or extending existing open-domain taxonomies such as Cyc [21] and WordNet [32]. The former results in a small taxonomy confined in a specific domain, and the latter enriches the taxonomy by adding more *named entities*. In other words, only the number of *instances* increases, not the number of *concepts* and the *isA* relationships between the concepts. More discussion can be found in [26].

Creating open-domain taxonomies from scratch is generally believed to be more challenging than the above tasks. The most notable work toward this effort is WikiTaxonomy [26] and YAGO [35]. Both of them attempt to derive a taxonomy[4] from the Wikipedia *categories*. Although these work reported high precision (not surprising due to the much cleaner data from Wikipedia), they have two fundamental limitations. First, Wikipedia categories are *thematic topics* used to classify Wikipedia articles, which are quite different from the terminology of *concepts* referred to by taxonomic definition. For instance, WikiTaxonomy contains "concepts" such as "*history of Athens*" or "*geology of Canada*", which are really topics and not concepts or categories. Second, the coverage of the taxonomies completely relies on the coverage of Wikipedia, which is still limited compared with Freebase or Probase.

To the best of our knowledge, there is no existing work on automatically inducing taxonomies in *web scale* similar to the effort of this paper. Prominent web-scale information extraction systems like KnowItAll [12] and TextRunner [2] *extract* the *isA* pairs from the web pages, but come short of constructing a taxonomy from these pairs with distinction on senses of the concepts.

4. A PROBABILISTIC TAXONOMY

Knowledge is not black and white. Removing all uncertainties is not only impossible but also harmful. Our philosophy is to live with noisy data and make the best use of it. In this section, we propose a probabilistic framework to model the knowledge in Probase. This framework essentially consists of two components: the joint probability of an *isA* pair, also known as the *plausibility*; and the conditional probability between a concept and its instances, also known as the *typicality*.

4.1 Plausibility

For every claim E in Probase we use $p(E)$ to denote the probability that it is true. In many cases, there is no clear distinction between true and false; and in almost all cases, information from external data sources is unreliable, inconsistent, or erroneous. We therefore interpret $p(E)$ as the *plausibility* of E. In this paper, we focus on *isA* relationships, that is, each claim E is a claim of an *isA* relationship between x and y, and $p(E) = \mathcal{P}(x, y)$.

To derive $p(E)$, we regard external information as evidence for E. Specifically, assume E is derived from a set of sentences or evidence $\{s_1, \cdots, s_n\}$ on the Web. Assume each piece of evidence s_i is associated with a probability p_i that reflects the belief or confidence of the evidence. Here, we adopt the simple noisy-or model. A claim E is false iff every piece of evidence in $s_1, \cdots, s_n$ is false. Since different sentences are extracted from different web pages, which are independently created by different people, we assume evidence is independent, and have

$$\mathcal{P}(x, y) = 1 - p(\bar{E}) = 1 - p(\bigwedge_{i=1}^{n} \bar{s_i}) = 1 - \prod_{i=1}^{n}(1 - p_i). \quad (1)$$

More sophisticated models (such as the Urns model [11]) can be used for plausibility. Due to lack of space, we focus our discussion on the noisy-or model. The model has good extensibility. It is easy to integrate new evidence, including *negative* evidence. A negative evidence claims that *A is not a B*, which effectively reduces the plausibility on the claim that *A is a B*. Example negative evidence includes the *part-of* relationship, e.g. "*B is comprised of A, C, and ...*" Incorporating a negative evidence s_i with probability p_i into plausibility is straightforward. We simply replace the factor $1 - p_i$ with p_i in Eq. (1).

A remaining issue is how to derive p_i for evidence s_i. We consider two factors. First, p_i may depend on the type of the information source (e.g. we consider evidence coming from the New York Times to be more credible than those from a public forum). Second, p_i may depend on how confident the information extraction process is when it identifies evidence s_i in the text. In our work, we characterize each s_i by a set of features F_i, such as: i) the PageRank score of the page from which s_i is extracted; ii) the Hearst pattern used in s_i; iii) the number of sentences with x as the super-concept; iv) the number of sentences with y as the sub-concept; v) number of sub-concepts in s_i; vi) position of y in s_i; and so on. Then, assuming independent features, we can apply Naive Bayes to derive p_i. Specifically, we have

$$p_i = p(s_i|F_i) = \frac{p(s_i) \cdot \prod_{f \in F_i} p(f|s_i)}{\sum_{s \in \{s_i, \bar{s_i}\}} p(s) \cdot \prod_{f \in F_i} p(f|s)}. \quad (2)$$

To learn the model, we use WordNet to build a training set. Given a pair (x, y), if both x and y appear in WordNet, and there is a path from x to y in WordNet, then we consider (x, y) as a *positive* example; if both x and y appear in WordNet, but there is no path from x to y, then we consider (x, y) as a *negative* example.

4.2 Typicality

Intuitively, a robin is more *typical* of the concept *bird* than is ostrich, while *Microsoft* is more *typical* of the concept *company* than is *Xyz Inc.*. However, in all existing taxonomies, instances inside a concept are treated *equally* typical. For example, the *company* concept in Freebase contains about 79,000 instances without a way to measure their typicality. Information about typicality is essential to many applications, such as those introduced in Section 1.

In this paper, we propose a typicality measure. Typicality exists in two directions: probability of an instance given a concept $\mathcal{T}(i|x)$ (a.k.a. instantiation); and probability of a concept given an instance $\mathcal{T}(x|i)$ (a.k.a. abstraction). We focus on the former here and the latter can be computed by Bayes rule in straightforward manner.

$\mathcal{T}(i|x)$ depends on two factors: i) the number $n(x, i)$ of evidence that supports the claim (x, i); and ii) the plausibility $\mathcal{P}(x, i)$ of the claim (x, i), as defined in Eq. (1). We then define the typicality of i to x as:

$$\mathcal{T}(i|x) = \frac{n(x, i) \cdot \mathcal{P}(x, i)}{\sum_{i' \in I_x} n(x, i') \cdot \mathcal{P}(x, i')}, \quad (3)$$

where I_x is the set of instances of the concept x.

One problem is that, besides the direct pair (x, i), i may also be an instance of a concept y which is a sub-concept of x. For example, besides the claim that *Microsoft* is an instance of the concept *company*, there are also claims in Probase that *Microsoft* is an instance of the concept *IT company* and *big company*. These claims

[4]YAGO is actually a more general ontology with taxonomic information included.

should also be regarded as additional evidence that *Microsoft* is a typical *company*.

To incorporate these *indirect* evidence as well, we refine Eq. (3) to be:

$$\mathcal{T}(i|x) = \frac{\sum_{y \in D(x)} \widetilde{\mathcal{P}}(x,y) \cdot n(y,i) \cdot \mathcal{P}(y,i)}{\sum_{i' \in I_x} \sum_{y \in D(x)} \widetilde{\mathcal{P}}(x,y) \cdot n(y,i') \cdot \mathcal{P}(y,i')}, \quad (4)$$

where $\widetilde{\mathcal{P}}(x,y)$ is the plausibility that y is a sub-concept or *descendant* concept of x, i.e., there exists at least one path from x to y. We use $D(x)$ to denote all the sub-concepts and descendant concepts of x. In particular, $x \in D(x)$ and we define $\widetilde{\mathcal{P}}(x,x) = 1$.

The remaining issue is to compute $\widetilde{\mathcal{P}}(x,y)$. Formally, let P_{xy} be the event that there is at least one path from x to y. Suppose $Parent(y)$ is the set of direct super-concepts of y. For each $z \in Parent(y)$, let P_z be the event that y is a direct sub-concept of z and there is at least one path from x to z, i.e. $P_z = P_{zy} \wedge P_{xz}$. Assuming independence of P_{zy} and P_{xz}, we have

$$p(P_z) = p(P_{zy} \wedge P_{xz}) = p(P_{zy}) \cdot p(P_{xz}) = \mathcal{P}(z,y) \cdot p(P_{xz}). \quad (5)$$

Furthermore, assuming independence of P_z for $z \in Parent(y)$, we then have

$$\begin{aligned} p(P_{xy}) &= p(\bigvee_{z \in Parent(y)} P_z) = 1 - p(\bigwedge_{z \in Parent(y)} \bar{P}_z) \\ &= 1 - \prod_{z \in Parent(y)} (1 - p(P_z)). \end{aligned} \quad (6)$$

Substituting Eq. (5) into Eq. (6), we obtain

$$\widetilde{\mathcal{P}}(x,y) = p(P_{xy}) = 1 - \prod_{z \in Parent(y)} (1 - \mathcal{P}(z,y) \cdot \widetilde{\mathcal{P}}(x,z)) \quad (7)$$

Algorithm 3: Computing $\widetilde{\mathcal{P}}(x,y)$

Input: T: the taxonomy
Output: $\Gamma = \{\widetilde{\mathcal{P}}(x,y)\}$: where there exists a path from x to y

```
1  Γ ← ∅;
2  k ← 1;
3  L^1 ← concepts of T with no parents;
4  while L^k ≠ ∅ do
5      foreach y ∈ L^k do
6          if Parent(y) = ∅ then
7              add P̃(y,y) = 1 to Γ;
8          else
9              foreach x ∈ Ancestor(y) do
10                 P̃(x,y) =
                   1 − ∏_{z∈Parent(y)} (1 − P(z,y) · P̃(x,z));
11                 add P̃(x,y) to Γ;
12             end
13         end
14     end
15     k ← k + 1;
16     L^k ← concepts of T that are not in ∪_{i=1}^{k−1} L^i but with all
       parents in ∪_{i=1}^{k−1} L^i;
17 end
18 return Γ;
```

Algorithm 3 describes a dynamic programming approach to compute $\{\widetilde{\mathcal{P}}(x,y)\}$. It traverses the taxonomy in a top-down fashion. Each time it computes some $\widetilde{\mathcal{P}}(x,y)$ at line 10, the required $\widetilde{\mathcal{P}}(x,z)$'s are guaranteed to have been computed. Deriving typicality from $\widetilde{\mathcal{P}}(x,y)$ is straightforward and thus not illustrated here.

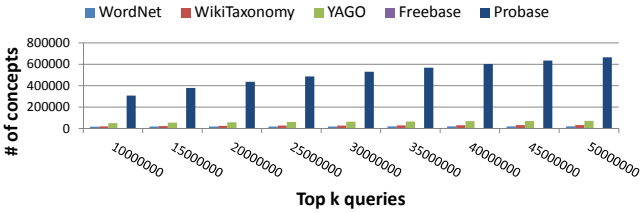


Figure 4: Number of relevant concepts in taxonomies

4.3 Related Work

Probabilistic approaches have been leveraged in some previous work [28, 32, 12, 2, 36]. In [28] and [32], frameworks based on statistical learning are used in taxonomy induction. In KnowItAll [12] and TextRunner [2], classifiers are used to assign confidence scores to the *isA* pairs extracted. Our work differs from these in two aspects. First, in previous work, probabilities are only used *during* the extraction or taxonomy induction stage, while the final output taxonomy is still *deterministic*. For instance, the confidence scores used in KnowItAll and TextRunner only serve the purpose of filtering incorrect *isA* pairs. Second, although the confidence scores in KnowItAll and TextRunner may share some similarity with the plausibility we defined here, the semantics of their scores are not clear. Furthermore, our focus on modeling typicality is novel and never explicitly addressed before. Although [36] leveraged scoring formulas (still *during* the induction phase) with a bit overlap in semantics as our typicality definition, their formulas are not applicable to general taxonomies, since they heavily rely on information specific to Wikipedia. They also did not take uncertainty (e.g., plausibility) into consideration, which we think is necessary for any automatic taxonomy inference framework. The effectiveness of the typicality in practical text understanding tasks has been demonstrated by our recent work [34, 39, 37].

5. EXPERIMENTAL EVALUATION

The proposed taxonomy inference framework was implemented on a cluster of servers using the *Map-Reduce* model. We used 7 hours and 10 machines to find all the *isA* pairs, and then 4 hours and 30 machines to construct the taxonomy. We also host Probase in a graph database system called Trinity [29, 30]. Due to space constraints, only highlights of the results are provided. Readers are referred to `http://research.microsoft.com/probase/` for complete experimental results.

We extract 326,110,911 sentences from a corpus containing 1,679,189,480 web pages. To the best of our knowledge, the scale of our corpus is one order of magnitude larger than the previously known largest corpus [27]. The inferred taxonomy has 2,653,872 distinct concepts, 16,218,369 distinct concept-instance pairs, and 4,539,176 distinct concept-subconcept pairs (20,757,545 pairs in total). Next we analyze the characteristics of the concept space and the *isA* relationship space of Probase, and briefly evaluate several applications that leverage typicality in conceptualization.

5.1 Concept Space

Given that Probase has more concepts than any other taxonomies, a reasonable question to ask is whether they are more effective in understanding text. We measure one aspect of the effectiveness here by examining Probase's concept coverage on web search queries. We define a concept to be *relevant*, if it appears at least once in web queries. We analyzed Bing's query log from a two-year period, sorted the queries in decreasing order by frequency,

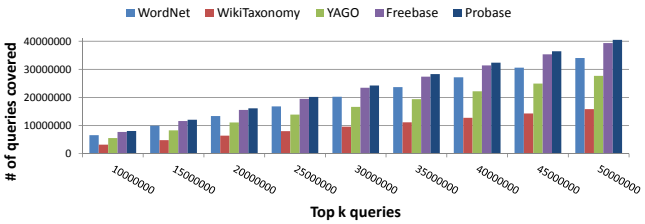


Figure 5: Taxonomy coverage of the top 50 million queries

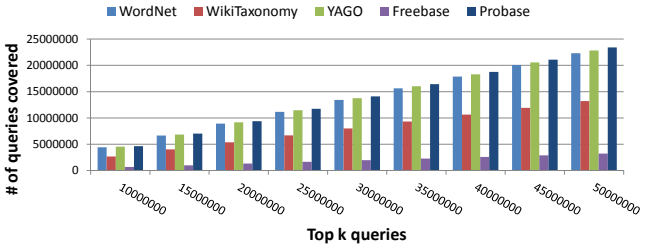


Figure 6: Concept coverage of the top 50 million queries

and computed the number of relevant concepts in Probase and four other general-purpose taxonomies, namely *WordNet*, *WikiTaxonomy* [26], *YAGO*, and *Freebase*, with respect to the top 50 million queries. Figure 4 shows the result.

In total, 664,775 concepts are considered relevant in Probase, compared to only 70,656 in YAGO. User web queries has a well-known *long-tail* distribution. While a small number of basic concepts (e.g., *company*, *city*, *country*) representing common sense knowledge appear very frequently in user queries, web users do mention other less known concepts. Probase does a better job at capturing these concepts in the long tail and hence has a better chance of understanding these user queries.

We next measure the *taxonomy coverage* of queries by Probase. A query is said to be *covered* by a taxonomy if the query contains at least one *concept or instance* within the taxonomy.[5] Figure 5 compares the coverage of queries by Probase taxonomy against four other taxonomies. Probase outperforms the others on the coverage of top 10 million to top 50 million queries. In all, Probase covers 40,517,506 (or, 81.04%) of the top 50 million queries.

We further measure *concept coverage*, which is the number of queries containing at least one *concept* in the taxonomy. Figure 6 compares the concept coverage by Probase against the other four taxonomies. Again, Probase outperforms all the others. Note that, although Freebase presents comparable taxonomy coverage with Probase in Figure 5, its concept coverage is much smaller.

In summary, with a larger concept space, Probase exhibits stronger ability in capturing the semantics implied in user queries. It is then expected that Probase can be a useful tool in interpreting these queries. Recent work [39] further demonstrates the effectiveness by leveraging Probase in query interpretation.

5.2 isA Relationship Space

There are two kinds of *isA* relationships in Probase: the concept-subconcept relationship which are the edges connecting internal nodes in the hierarchy, and the concept-instance relationship which are the edges connecting a leaf node.

Table 4 compares the concept-subconcept relationship space of Probase with the other taxonomies. The *level* of a concept is defined to be the length of the longest path from it to a leaf node (i.e. an instance). Table 4 shows that even with an order of magnitude larger number of concepts, Probase still has a comparable hierarchical complexity to the other taxonomies. The exception is Freebase which exhibits trivial values on these measured metrics because it has no *isA* relationship among its concepts at all.

[5]We say "cover" to mean that the taxonomy *contributes* to the understanding of the query as it understands at least one term in the query.

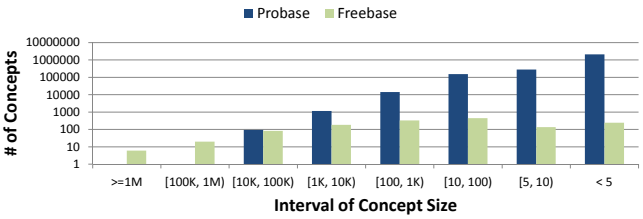


Figure 7: Concept size distributions in Probase and Freebase

	# of *isA* pairs	Avg # of children	Avg # of parents	Avg level	Max level
WordNet	283,070	11.0	2.4	1.265	14
WikiTaxonomy	90,739	3.7	1.4	1.483	15
YAGO	366,450	23.8	1.04	1.063	18
Freebase	0	0	0	1	1
Probase	4,539,176	7.53	2.33	1.086	7

Table 4: The concept-subconcept relationship space

We also compare Probase and Freebase on the concept-instance relationships. We choose Freebase since it is the only existing taxonomy with comparable scale on instance space (24,483,434 concept-instance pairs). We define *concept size* to be the number of instances directly under a concept node.

Figure 7 compares distributions of concept sizes in Probase and Freebase. While Freebase focuses on a few very popular concepts like "*track*" and "*book*" with over two million instances, Probase has many more medium to small sized concepts. In fact, the top 10 concepts in Freebase contain 17,174,891 concept-instance pairs, or 70% of all the pairs it has. In contrast, the top 10 concepts in Probase only contain 727,136 pairs, or 4.5% of its total. Therefore, Probase provides a much broader coverage on diverse topics, while Freebase is more informative on specific topics. On the other hand, the instances of large concepts in Freebase like *book* are mostly from specific web sites like Amazon, which can be easily merged into Probase.

To estimate the correctness of the extracted *isA* pairs in Probase, we create a benchmark dataset of 40 concepts in various domains. Due to space limitations, we show 20 of them in Table 5. The complete benchmark can be found in the longer version of this paper [40]. The concept size varies from 21 instances (for *aircraft model*) to 85,391 (for *company*), with a median of 917. Benchmarks with similar number of concepts and domain coverage have also been reported in information extraction research [25]. For each concept, we randomly pick up to 50 instances/subconcepts and ask human judges to evaluate their correctness.

Figure 8 shows the result. The average precision of all pairs in benchmark is 92.8%, which outperforms precision reported from other previous notable information extraction frameworks like KnowItAll [12] (64% on average), NELL [7] (74%) and TextRunner [2] (80% on average). It is not fair to directly compare our results with Wikipedia-based frameworks like WikiTaxonomy [26] (86%) and YAGO [35] (95%), whose data sources are much cleaner. Nevertheless, only YAGO has a better overall precision than Probase.

We also examined each iteration of the extraction framework. Figure 9 shows the accumulated number of *isA* pairs and concepts extracted after each round of iteration. Both curves grow up quickly in the first several rounds and then saturates as the bootstrapping process converges. An interesting phenomenon is that the largest gain actually occurs in the second round instead of the first one. This is because in the first round, the ambiguity in many sentences

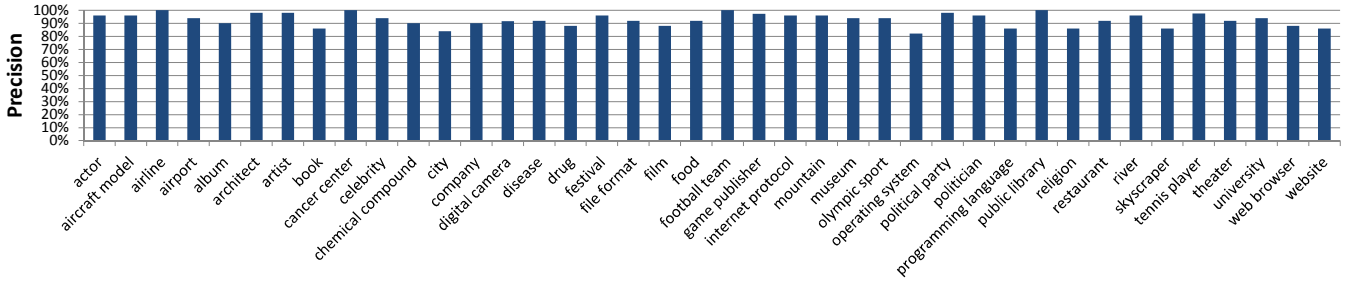


Figure 8: Precision of extracted pairs

Concept (# of Instances)	Typical Instances
actor (3466)	Tom Hanks, Marlon Brando, George Clooney
aircraft model (21)	Airbus A320-200, Piper PA-32, Beech-18
airline (1221)	British Airways, Deltae
album (1938)	Thriller, Big Calm, Dirty Mind
cancer center (55)	Fox Chase, Care Alliance, Dana-Farber
chemical compound (308)	carbon dioxide, phenanthrene, carbon monoxide
company (85391)	IBM, Microsoft, Google
disease (8784)	AIDS, Alzheimer, chlamydia
festival (3039)	Sundance, Christmas, Diwali
film (13402)	Blade Runner, Star Wars, Clueless
football team (59)	Real Madrid, AC Milan, Manchester United
internet protocol (168)	HTTP, FTP, SMTP
museum (2441)	the Louvre, Smithsonian, the Guggenheim
operating system (86)	Linux, Solaris, Microsoft Windows
politician (953)	Barack Obama, Bush, Tony Blair
public library (39)	Haringey, Calcutta, Norwich
restaurant (4546)	Burger King, Red Lobster, McDonalds
skyscraper (121)	the Empire State Building, the Sears Tower, Burj Dubai
theater (632)	Metro, Pacific Place, Criterion
web browser (232)	Internet Explorer, Firefox, Safari

Table 5: 20 Benchmark concepts and their typical instances

cannot be resolved given a growing but limited Γ. In the second round, we can leverage a more comprehensive Γ to extract more pairs from those previously unsolved sentences.

Figure 10 further shows the average precision of the *isA* pairs extracted for the 40 benchmark concepts after each round of iteration. The precision decreases slightly from 97.3% as the iteration goes on. The main reason for the decrease is that, as the iteration proceeds, although the number of correct pairs in Γ increases, so does the number of *incorrect* ones. This can cause problems in super-concept detection. For example, consider the sentence "... non-caged animals other than dogs **such as** *cats*, *wolves*, *fish* ...". It is possible that at one point Γ contains both the pair (dogs, cats) and (non-caged animals, dogs). Going forward, as the frequency of (dogs, cats) increases, the frequency of (non-caged animals, cats) may remain unchanged due to the rare occurrence of "non-caged animals" in corpus. Therefore it is possible for the likelihood ratio $r(dogs, non\text{-}caged\ animals)$ to exceed the threshold after some time, and then the algorithm will mistake "dogs" as the correct super-concept. Consequently, other incorrect pairs like (dogs, wolves) and (dogs, fish) will be introduced in Γ, which further degrades the precision.

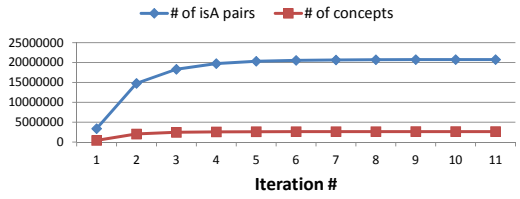


Figure 9: The number of *isA* pairs and concepts extracted

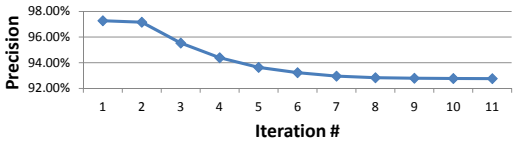


Figure 10: Precision of *isA* pairs extracted

5.3 Applications

Typicality has also been leveraged in several recent Probase-enabled applications, such as *taxonomy keyword search* [9], *semantic web search* [39], *short text understanding* [34, 38], and *understanding web tables* [37]. The reported results show that, the typicality modeled in Section 4.2 can help to either address important problems not touched before [39, 37], or enhance performance for approaches on existing text understanding tasks (e.g., [34]). We next briefly describe two of these applications: *semantic web search* and *short text understanding*. They illustrate how typicality is used in the two kinds of conceptualization tasks outlined in Section 1, i.e., *instantiation* and *abstraction*, respectively.

Semantic Web Search. As shown in Section 5.2, more than 80% of Web searches contain concepts and/or instances that can be found in Probase. This gives Probase a good advantage in interpreting user intents. Consider the following search queries: i) *ACM fellows working on semantic web*; and ii) *database conferences in asian cities*. The user intents of these queries are clear. However, current keyword-based search engines cannot deliver good results as they return pages with exact, word-for-word matches for phrases such as "ACM fellows", "database conferences", and "asian cities".

We build a novel semantic search prototype to handle these semantic queries [39]. Due to the large coverage on concepts in Probase, we can easily identify the concepts within user queries. We then rewrite the queries by substituting the concepts with their most typical instances by typicality scores. For example, we can replace *database conferences* with *SIGMOD*, *VLDB*, *ICDE*, etc., and replace *Asian cities* with *Beijing*, *Singapore*, *Tokyo*, etc.. The refined queries become *SIGMOD in Beijing*, etc. Since there are many possible combinations of two instances for the two concepts, we use *word association* [39] between instances and key words to determine the best pair of instances for substitution. Our evaluation shows that, on average, about 80% of the returned results from those rewritten queries are considered as relevant by users,

compared with less than 50% of those results from searching the original queries on Google or Bing.

Short Text Understanding. Understanding short text (e.g. web search, tweets, anchor texts) is important to many applications. Statistical approaches such as *topic models* [3] treat text as a *bag of words*, and discover *latent topics* from the text. However, finding latent topics is not tantamount to understanding. A latent topic is represented by a set of words, consisting of no *explicit* concepts. Further more, bag-of-words approaches typically do not work well for short texts which do not have enough signals.

Probase enables machines to conceptualize from a set of words by performing Bayesian analysis based on the typicality $\mathcal{T}(x|i)$. For example, given a word *India*, the machine can infer its most *typical* concepts such as *country* or *region*. Given two words, *India* and *China*, the most typical concepts become *Asian country* or *developing country*, etc. Adding another word, *Brazil*, the top concepts may become *BRIC* or *emerging market*, etc. In a related work [34], we generalize this conceptualization approach to general *terms* (not just words that are instances). We then cluster twitter messages based on conceptual signals (and their typicality scores) provided by Probase. We represent each tweet as a vector with its most typical concepts as features, and perform K-means clustering. The results outperform all existing approaches such as LDA [3].

6. CONCLUSION

In this paper, we presented a framework which automatically inferences an open-domain, probabilistic taxonomy from the entire web. This taxonomy, to the best of our knowledge, is currently the largest and the most comprehensive in terms of the number of concepts included. Its probabilistic model allows the integration of both precise and ambiguous knowledge and even tolerates inconsistencies and errors which are common on the Web. More importantly, this model enables probabilistic inference between concepts and instances which will benefit a wide range of applications that require text understanding.

7. REFERENCES

[1] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. Dbpedia: A nucleus for a web of open data. In *ISWC/ASWC*, pages 722–735, 2007.

[2] M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. Open information extraction from the web. In *IJCAI*, pages 2670–2676, 2007.

[3] D. Blei and J. Lafferty. Topic models. In *Text Mining: Classification, Clustering, and Applications*. Taylor & Francis, 2009.

[4] P. Bloom. Glue for the mental world. *Nature*, 421:212–213, Jan 2003.

[5] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*, 2008.

[6] S. A. Caraballo. Automatic construction of a hypernym-labeled noun hierarchy from text. In *ACL*, 1999.

[7] 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 *AAAI*, 2010.

[8] P. Cimiano, A. Pivk, L. S. Thieme, and S. Staab. Learning taxonomic relations from heterogeneous sources of evidence. In *Proceedings of the ECAI 2004 Ontology Learning and Population Workshop*, 2004.

[9] B. Ding, H. Wang, , R. Jin, J. Han, and Z. Wang. Optimizing index for taxonomy keyword search. In *SIGMOD*, 2012.

[10] D. Downey, M. Broadhead, and O. Etzioni. Locating complex named entities in web text. In *IJCAI*, pages 2733–2739, 2007.

[11] D. Downey, O. Etzioni, and S. Soderland. A probabilistic model of redundancy in information extraction. In *IJCAI*, 2005.

[12] 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*, pages 100–110, 2004.

[13] C. Fellbaum, editor. *WordNet: an electronic lexical database*. MIT Press, 1998.

[14] M. Fleischman. Automated subcategorization of named entities. In *ACL (Companion Volume)*, pages 25–30, 2001.

[15] M. Fleischman and E. H. Hovy. Fine grained classification of named entities. In *COLING*, 2002.

[16] M. A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *COLING*, pages 539–545, 1992.

[17] T. Lee, Z. Wang, H. Wang, and S. Hwang. Web scale taxonomy cleansing. In *VLDB*, 2011.

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

[19] P. Li, H. Wang, H. Li, and X. Wu. Sparse information extraction based on semantic contexts. Under submission, 2012.

[20] Z. Li, H. Li, H. Wang, and X. Zhou. Overcoming semantic drift in web-scale information extraction. Under submission, 2012.

[21] C. Matuszek, M. J. Witbrock, R. C. Kahlert, J. Cabral, D. Schneider, P. Shah, and D. B. Lenat. Searching for common sense: Populating cyc from the web. In *AAAI*, pages 1430–1435, 2005.

[22] G. Murphy. *The big book of concepts*. The MIT Press, 2004.

[23] D. Nadeau and S. Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007.

[24] R. Navigli. Word sense disambiguation: A survey. *ACM Comput. Surv.*, 41(2), 2009.

[25] M. Pasca. Organizing and searching the world wide web of facts – step two: harnessing the wisdom of the crowds. In *WWW*, 2007.

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

[27] A. Ritter, S. Soderland, and O. Etzioni. What is this, anyway: Automatic hypernym discovery. In *AAAI Spring Symposium on Learning by Reading and Learning to Read*, 2009.

[28] E. Segal, D. Koller, and D. Ormoneit. Probabilistic abstraction hierarchies. In *NIPS*, pages 913–920, 2001.

[29] B. Shao, H. Wang, and Y. Li. The Trinity graph engine. Technical report, Microsoft Research, 2012.

[30] B. Shao, H. Wang, and Y. Xiao. Managing and mining large graphs: Systems and implementations. In *SIGMOD*, 2012.

[31] P. Singh, T. Lin, E. Mueller, G. Lim, T. Perkins, and W. Li Zhu. Open Mind Common Sense: Knowledge acquisition from the general public. *On the Move to Meaningful Internet Systems: CoopIS, DOA, and ODBASE*, pages 1223–1237, 2002.

[32] R. Snow, D. Jurafsky, and A. Y. Ng. Semantic taxonomy induction from heterogenous evidence. In *ACL*, 2006.

[33] R. Snow, S. Prakash, D. Jurafsky, and A. Y. Ng. Learning to merge word senses. In *EMNLP-CoNLL*, pages 1005–1014, 2007.

[34] Y. Song, H. Wang, Z. Wang, H. Li, and W. Chen. Short text conceptualization using a probabilistic knowledgebase. In *IJCAI*, 2011.

[35] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: a core of semantic knowledge. In *WWW*, pages 697–706, 2007.

[36] C. Thomas, P. Mehra, R. Brooks, and A. P. Sheth. Growing fields of interest - using an expand and reduce strategy for domain model extraction. In *Web Intelligence*, pages 496–502, 2008.

[37] J. Wang, Z. Wang, H. Wang, and K. Q. Zhu. Understanding tables on the web. Technical report, Microsoft Research, 2010.

[38] S. Wang, Y. Song, H. Wang, and Z. Zhang. On understanding short texts. Under submission, 2012.

[39] Y. Wang, H. Li, H. Wang, and K. Q. Zhu. Toward topic search on the web. Technical report, Microsoft Research, 2010.

[40] W. Wu, H. Li, H. Wang, and K. Q. Zhu. Probase: A probabilistic taxonomy for text understanding. Technical report, Microsoft Research, 2012.