DATA PAPER

Microsoft Concept Graph: Mining Semantic Concepts for Short Text Understanding

Lei Ji[1,2†], Yujing Wang[1], Botian Shi[3], Dawei Zhang[4], Zhongyuan Wang[5] & Jun Yan[6]

[1]Microsoft Research Asia, Haidian District, Beijing 100080, China

[2]Institute of Computing Technology, Chinese Academy of Sciences, Haidian District, Beijing 100049, China

[3]Beijing Institute of Technology, Haidian District, Beijing 100081, China

[4]MIX Labs, Haidian District, Beijing 100080, China

[5]Meituan NLP Center, Chaoyang District, Beijing 100020, China

[6]AI Lab of Yiducloud Inc., Huayuan North Road, Haidian District, Beijing 100089, China

Keywords: Knowledge extraction; Conceptualization; Text understanding

Citation: L. Ji, Y. Wang, B. Shi, D. Zhang, Z. Wang & J. Yan. Microsoft concept graph: Mining semantic concepts for short text understanding. Data Intelligence 1(2019). doi: 10.1162/dint_a_00013
Received: November 20, 2018; Revised: February 12, 2019; Accepted: February 19, 2019

ABSTRACT

Knowlege is important for text-related applications. In this paper, we introduce Microsoft Concept Graph, a knowledge graph engine that provides concept tagging APIs to facilitate the understanding of human languages. Microsoft Concept Graph is built upon Probase, a universal probabilistic taxonomy consisting of instances and concepts mined from the Web. We start by introducing the construction of the knowledge graph through iterative semantic extraction and taxonomy construction procedures, which extract 2.7 million concepts from 1.68 billion Web pages. We then use conceptualization models to represent text in the concept space to empower text-related applications, such as topic search, query recommendation, Web table understanding and Ads relevance. Since the release in 2016, Microsoft Concept Graph has received more than 100,000 pageviews, 2 million API calls and 3,000 registered downloads from 50,000 visitors over 64 countries.

† Corresponding author: Lei Ji (Email: leiji@microsoft.com; ORCID: 0000-0002-7569-3265).

1. INTRODUCTION

Concepts are indispensable for humans and machines to understand the semantic meanings underlined in the raw text. Humans understand an instance, especially an unfamiliar instance, by its basic concept in an appropriate level, which is defined as Basic-level Categorization (BLC) by psychologists and linguists. For example, people may not understand "Gor Mahia", but with the concept "football club" described in Wikipedia, people can capture the semantic meaning easily. Psychologist Gregory Murphy began his highly acclaimed book [1] with the statement "Concepts are the glue that holds our mental world together". *Nature* magazine book review [2] calls it an understatement, because "Without concepts, there would be no mental world in the first place". To enable machines to understand the concept of an instance like human beings, one needs a knowledge graph consisted of instances, concepts, as well as their relations. However, we observe two major limitations in existing knowledge graphs, which motivate us to build a brand-new knowledge taxonomy, Probase [3], to tackle general purpose understanding of human language.

1). **Previous taxonomies have limited concept space.** Many existing taxonomies are constructed by human experts and are difficult to be scaled up. For example, Cyc project [4] contains about 120,000 concepts after 25 years of evolution. Some other projects, like Freebase [5], resort to crowd sourcing efforts to increase the concept space, which still lacks general coverage of many other domains and thus holds a barrier for general purpose text understanding. There are also automatically constructed knowledge graphs, such as YAGO [6] and NELL [7]. Nevertheless, the coverages of these concept spaces are still limited. The number of concepts of Probase and some other popular open-domain taxonomies are shown in Table 1, which demonstrates that Probase has a much larger concept space.

Table 1. Concept space comparison of existing taxonomies.

Existing taxonomies	Number of concepts
Freebase [5]	1,450
WordNet [8]	25,229
WikiTaxonomy [9]	111,654
YAGO [6]	352,297
DBPedia [10]	259
Cyc [4]	≈120,000
NELL [7]	123
Probase [3]	**≈5,400,000**

2). **Previous taxonomies treat knowledge as black and white.** Traditional knowledge base aims at providing standard, well-defined and consistent reusable knowledge, and treats knowledge as black and white. However, treating knowledge as black and white obviously has restrictions, especially when the concept space is extremely large, because different knowledge has different confidence intervals or probabilities, and the best threshold of probabilities depends on the specific application. Different from previous taxonomies, our philosophy is to annotate knowledge facts with probabilities and let the application itself decide the best way of using it. We believe this design is more flexible and can be utilized to benefit a broader range of text-based applications.

Based on the Probase knowledge taxonomy, we propose a novel conceptualization model to learn text representation in the Probase concept space. The conceptualization model (also known as the Concept Tagging Model) aims to map text into semantic concept categories with some probabilities. It provides computers the commonsense computing capability and makes machines "aware" of the mental world of human beings, through which machines can better understand human communication in text. Based on the conceptualization model, the Probase taxonomy can be applied to facilitating various text-based applications, including topic search, query recommendation, Ads relevance, Web table understanding, etc.

An overview of the entire framework is illustrated in Figure 1, which mainly consists of three layers: (1) knowledge construction layer, (2) conceptualization layer, and (3) application layer.

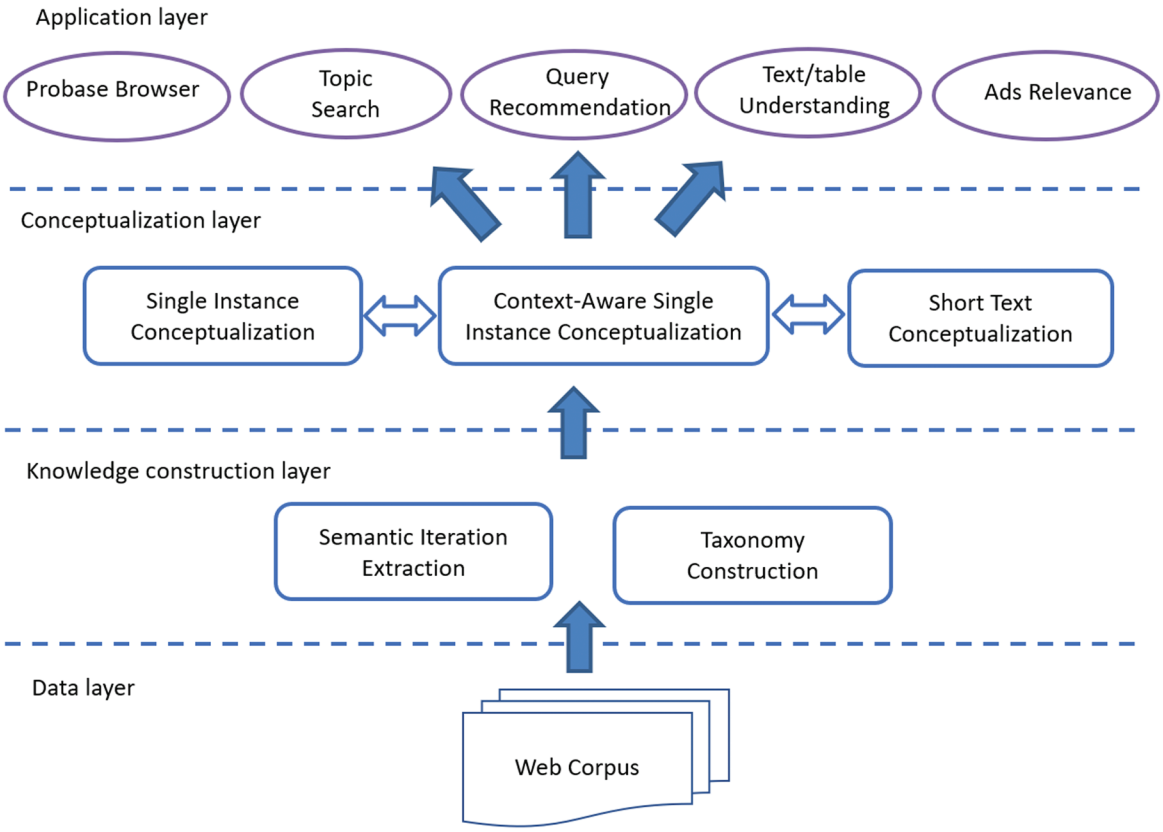


Figure 1. The framework overviews.

1). **Knowledge construction layer.** In the knowledge construction layer, we address the construction of Probase knowledge network. First, the isA facts are mined automatically from the Web via a semantic iteration extraction procedure. Second, the taxonomy is constructed following a rule-based taxonomy construction algorithm. Finally, we calculate the probability score for each related instance/concept in the taxonomy.

2). **Conceptualization layer**. Based on the constructed sematic knowledge network, we design a conceptualization model to represent raw text in the Probase concept space. The model can be divided into three major components, namely (1) single instance conceptualization, (2) context-aware single instance conceptualization, and (3) short text conceptualization.
3). **Application layer**. The conceptualization model enables us to generate "Bag-of-Concepts" representation of raw text, which can be utilized to enhance various categories of applications, including but not limited to topic search, query recommendation, Ads relevance calculation and Web table understanding. We also build a Probase browser and a tagging model demo in the application layer, which provide users a quick insight into a specific query.

The rest of this paper is organized as follows. Section 2 discusses related works, and Section 3 presents the construction of the Probase semantic network. Section 4 introduces the conceptualization model built upon the sematic network, and Section 5 shows some example applications. In addition, Section 6 lists the data sets and statistics. Finally, we make a conclusion in Section 7.

2. RELATED WORKS

Knowledge graph has attracted great interests in many research fields. There are many existing knowledge graphs built either manually or automatically.

1). **WordNet** [8] Different from traditional thesauruses, WordNet groups words together based on their meanings instead of morphology. Terms are grouped into synsets, where each term represents a distinct concept. Synsets are interlinked by conceptual semantics and lexical relations. WordNet has more than 117,000 synsets for 146,000 terms. WordNet is widely used for term disambiguation as it defines various senses for a term.
2). **DBpedia** [10] is a project of extracting Wikipedia Infobox into knowledge facts which can be semantically queried using SPARQL. The knowledge in DBpedia is extracted from Wikipedia and collaboratively edited by the community. DBpedia has released 670 million triples in RDF format. DBpedia provides an ontology including 280 classes, 3.5 million entities and 9,000 attributes.
3). **YAGO** [6], abbreviation for Yet Another Great Ontology, is an open sourced huge semantic knowledge base, which has fused knowledge extracted from WordNet, GeoNames[①] and Wikipedia[②]. YAGO combines the taxonomy of WordNet with the Wikipedia category to assign entities to more than 200,000 classes. YAGO is comprised by more than 120 million facts about 3 million entities. Based on manual evaluation, the accuracy of YAGO is about 95%. YAGO also attaches temporal and special information to the entities and relations by some declarative extraction rules.

[①] http://www.geonames.org/
[②] https://www.wikipedia.org/

4). **Freebase** [5] is a large knowledge base containing a lot of human labeled data contributed by community members. Freebase extracts data from many sources including Wikipedia, NNDB[③], Fashion Model Directory[④] and MusicBrainz[⑤]. Freebase has more than 125 million facts, 4,000 types and 7,000 properties. After 2016, all data in Freebase have been transferred to Wikidata[⑥].
5). **ConceptNet** [11] is a semantic network aiming to build a large commonsense knowledge base in a crowd sourcing manner, which focuses on the commonsense relations including isA, partOf, usedFor and capableOf. ConceptNet contains over 21 million edges and over 8 million nodes.
6). **NELL** [7] is a continuously running system which extracts facts from text in hundreds of millions of Web pages in an iterative way, while patterns are boosted during the process. NELL has accumulated more than 50 million candidate beliefs and has extracted 2,810,379 asserted instances of 1,186 different categories and relations.
7). **WikiTaxonomy** [9] presents a taxonomy automatically generated from the categories in Wikipedia, which generates 105,418 isA links from a network of 127,325 categories and 267,707 links. The system achieves 87.9 balanced *F*-measure according to a manual evaluation on the taxonomy.
8). **KnowItAll** [12] aims to automate the process of extracting large collections of facts from the Web in an unsupervised, domain-independent and scalable manner. The system has three major components: Pattern Learning (PL), Subclass Extraction (SE) and List Extraction (LE), achieving great improvements on the recall while maintaining precision and enhancing the extraction rate.

The existing knowledge graphs suffer from low coverage and lack of well-organized taxonomies. Moreover, none of them focuses on extracting the super-concepts and sub-concepts of an instance. To automatically generate the taxonomy, we propose Probase which automatically constructs the semantic network of isA facts.

3. SEMANTIC NETWORK CONSTRUCTION

The entire data construction procedure of the Probase semantic network will be introduced in detail in the following subsections. First, in Section 3.1, we describe the iterative data extraction process; then, the taxonomy construction step is introduced in Section 3.2; Section 3.3 introduces the algorithm of probability calculation.

3.1 Iterative Extraction

The Probase semantic network [3] is built upon isA facts extracted from the Web. The isA facts can be formulated as pairs consisting of a super-concept and a sub-concept. For example, "cat isA animal" forms a pair, where "cat" is the sub-concept and "animal" is the super-concept. In this work, we propose a method based on semantic iteration to mine the isA pairs from Web.

③ http://www.nndb.com/
④ https://www.fashionmodeldirectory.com/
⑤ https://musicbrainz.org
⑥ https://www.wikidata.org

3.1.1 Syntactic vs. Semantic Iteration

Before our work, the state-of-the-art information extraction methods [7, 8, 12] rely on a syntactic integrative (bootstrapping) approach. It starts with a set of seed patterns to discover example pairs with high confidence. Then, we can derive new patterns from the extracted pairs and use the new patterns to extract more pairs. The iterative process is continued until a certain stop criterion is reached. However, we observe in practice that the syntactic iteration methods have indispensable barriers in deep knowledge acquisition because high quality syntactic patterns are rare. Therefore, we propose a semantic interactive approach by which the new pairs can be extracted with high confidence based on knowledge accumulated from existing pairs.

3.1.2 The Semantic Iteration Algorithm

First, we extract a set of candidate pairs by Hearst patterns [13] (Table 2). For instance, if we have sentence "… animals **such as** cat …", we can extract a pair <animal, cat> which means cat isA animal. Sometimes, there is ambiguity in the pattern matching process. For instance, in the sentence "… animals **other than** dogs **such as** cats …", we can extract two candidate pairs <animals, cats>, and <dogs, cats>. The algorithm must have the ability to decide which one is the correct super-concept among "animals" and "dogs". Another example is "… companies **such as** IBM, Nokia, Proctor and Gamble …". In this case, we have multiple choices in the sub-concept, namely, <companies, Proctor> and < companies, Proctor and Gamble>. Again, the algorithm should automatically determine their correctness.

Table 2. Hearst patterns examples.

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

We denote the candidate set of super-concepts for sentence s as X_s, and the candidate set of sub-concepts for sentence s as Y_s. Γ is the set of isA pairs that have already been extracted as ground truth. The remaining task for the algorithm is to detect the correct super-concepts and sub-concepts from X_s and Y_s, respectively, based on the knowledge accumulated in Γ. For each sentence s, if we have multiple choices for the super-concepts, we must choose one as the correct super-concept. It is based on the observation that when ambiguity exists in super-concept matching, there is only one correct answer. After determining the correct super-concept, the goal is to filter out the correct sub-concepts from candidates in Y_s. Unlike the super-concept case, we may expect more than one valid sub-concept, and the result sub-concepts should be a subset of Y_s.

3.1.3 Super-Concept Detection

Let $X_s = \{x_1, x_2, \ldots, x_m\}$, we compute likelihood $p(x_k|Y_s)$ for each candidate x_k. We assume $y_1, y_2, \ldots, y_n \in Y_s$ are independent with each other given any super-concept x_k, and then we have

$$p(x_k|Y_s) \propto p(x_k)p(Y_s \mid x_k) = p(x_k)\prod_{i=1}^{n} p(y_i \mid x_k), \tag{1}$$

where $p(x_k)$ is the percentage of pairs in Γ that contain x_k as the super-concept, and $p(y_i|x_k)$ is the percentage of pairs in Γ that have y_i as the sub-concept when the super-concept is x_k. There are pairs that do not exist in Γ, and therefore, we let $p(y_i|x_k) = \varepsilon$ if no existence can be found for that pair; where ε is set to be a small positive number. Without losing generality, we assume x_1 and x_2 are the top two candidates with the highest probability scores, and $p(x_1|Y_s) > p(x_2|Y_s)$. Then, we can compute the ratio of two probabilities as follows:

$$r(x_1, x_2) = \frac{p(x_1)\prod_{i=1}^{n} p(y_i \mid x_1)}{p(x_2)\prod_{i=1}^{n} p(y_i \mid x_2)}, \tag{2}$$

x_1 will be chosen as the correct super-concept if $r(x_1,x_2)$ is larger than a pre-defined threshold. If not, this sentence is skipped in the current iteration and may be recovered in a later round when Γ is more informative. Intuitively, the likelihood p (animals|cats) should be much larger than p (dogs|cats); so, we can correctly select "animals" as the result super-concept.

3.1.4 Sub-Concept Detection

In our implementation, sub-concept detection is based on the features extracted from the original sentences, which is motivated by two basic observations.

Observation 1: The closer a candidate sub-concept is to the pattern keyword, the more likely it is a valid sub-concept.

Observation 2: If we are certain that a candidate sub-concept at the k-th position (calculated by the distance from the pattern keyword) is valid, the candidate sub-concepts from position 1 to position k–1 are also likely to be correct.

For example, consider the following sentence: "… representatives in North America, Europe, the Middle East, Australia, Mexico, Brazil, Japan, China, **and other** countries …". Here "**and other**" is the pattern keyword; *China* is the candidate sub-concept closest to the pattern keyword. Obviously, *China* is a correct sub-concept. In addition, if we know that *Australia* is a legal sub-concept of "*countries*", we can be more confident that *Mexico*, *Brazil* and *Japan* are all correct sub-concepts of the same category; while *North America*, *Europe* and *the Middle East* are less likely to be the instances of the same category.

Specifically, we find the largest k such that the likelihood $p(y_k|x)$ is above a pre-defined threshold. Then, y_1, y_2,…, y_k are all treated as valid sub-concepts of the super-concept x. Note that there are sometimes ambiguity in the sub-concept y_i. For instance, in the sentence "… companies **such as** IBM, Nokia, Proctor

and Gamble…" at position 3, the noun phrase can be "Proctor" or "Proctor and Gamble". Suppose the candidate at the position k to be c_1, c_2,…, we calculate the conditional probability of each candidate given the super-concept x as well as all sub-concepts before position k:

$$p(y_k = c_i|x, y_1, y_2, \ldots, y_{k-1}) = p(c_i|x)\prod_{j=1}^{k-1} p(y_j \mid c_i, x), \tag{3}$$

As before, y_1, y_2,…, y_{k-1} are assumed to be independent given the super-concept x; $p(c_i|x)$ is the percentage of existing pairs in Γ where c_i is a sub-concept when x is the super-concept; $p(y_j|c_i, x)$ is the likelihood that y_j is a valid sub-concept given x is the super-concept and c_i is another valid sub-concept in the same sentence. Suppose c_1 and c_2 are the top 2 ranked concepts, and we pick c_1 as the final concept if $r(c_1, c_2)$ exceeds a certain ratio. The value of $r(c_1, c_2)$ can be calculated as:

$$r(c_1, c_2) = \frac{p(c_1 \mid x)\prod_{j=1}^{k-1} p(y_j \mid c_1, x)}{p(c_2 \mid x)\prod_{j=1}^{k-1} p(y_j \mid c_2, x)}, \tag{4}$$

Intuitively, the probability p(Proctor and Gamble|companies) should be much larger than p(Proctor|companies) after Γ accumulates enough knowledge, and thus the algorithm is able to select the correct candidate automatically.

3.2 Taxonomy Construction

The taxonomy construction procedure mainly consists of three steps, (1) local taxonomy construction, (2) horizontal merge and (3) vertical merge. First, we build the local taxonomy for each sentence. Then, we perform horizontal merge and vertical merge in sequence on the local taxonomy collection to construct a global taxonomy.

1). **Local Taxonomy Construction.** Figure 2 illustrates the process to build a local taxonomy from each sentence. For example, in the sentence "A **such as** B, C, D", we can get three pairs <A, B>, <A, C>, and <A, D> where A is the super-concept and B, C, D are the sub-concepts. We can build a local taxonomy with A as the root node and B, C, D as child nodes.

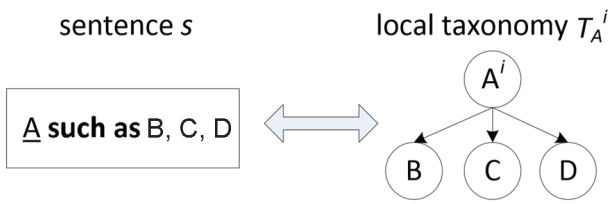


Figure 2. Local taxonomy construction.

2). **Horizontal Merge.** After the local taxonomy for each sentence is built, the next step is horizontal merge. Suppose we have two local taxonomies T_A^1 and T_A^2 whose root is A^1 and A^2, respectively, where A^1 and A^2 correspond to the same surface form. If we are sure that A^1 and A^2 express the same sense, we can merge the two trees horizontally, as shown in Figure 3. We design a similarity function to calculate the probability that A^1 and A^2 are semantically equal. Intuitively, if the children of A^1 and A^2 are more overlapped, we can be more confident that they are of the same sense. Therefore, we adopt absolute overlap for the similarity calculation function, i.e., $f\left(A^1, A^2\right) = \left|A^1 \cap A^2\right|$; and a constant threshold δ is used to determine whether two local taxonomies can be horizontally merged.

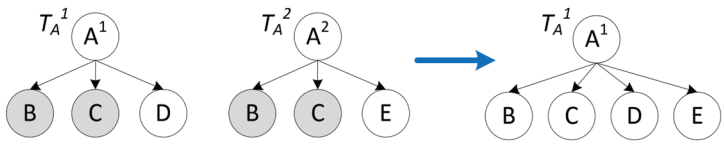


Figure 3. Horizontal merge.

3). **Vertical Merge.** Given two local taxonomies rooted by A^i and B^1, where B is a child of A^i. If we are confident that B is of the same sense as B^1, we can merge the two taxonomies vertically. As shown in Figure 4, after merging, A^i is the root; the original subtree B is merged with the taxonomy rooted by B^1; and the other subtrees C and D remain at the same position. To determine if B and B^1 are of the same sense, we calculate the absolute overlap of R^1's children and B's siblings (emphasized by colored nodes in Figure 4).

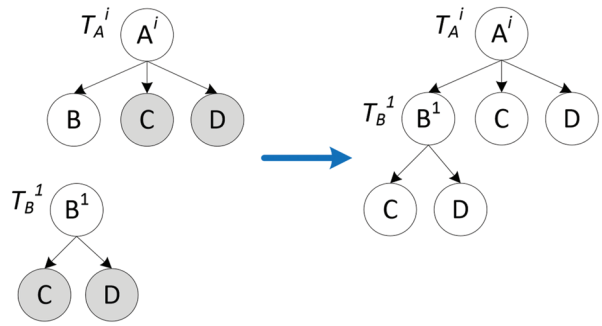


Figure 4. Vertical merge: single sense alignment.

There is another case which is more complicated (Figure 5). Both T_B^1 and T_B^2 can be vertically merged with T_A^i, as the child nodes of R^1 and R^2 have considerable overlap with B's siblings in T_A^i. However, the child nodes of R^1 and R^2 do not have enough overlap, indicating that R^1 and R^2 may express different senses. In this case, we split two senses in the merged taxonomy, i.e., T_B^1 and T_B^2 are merged as two distinct subtrees in taxonomy T_A^i.

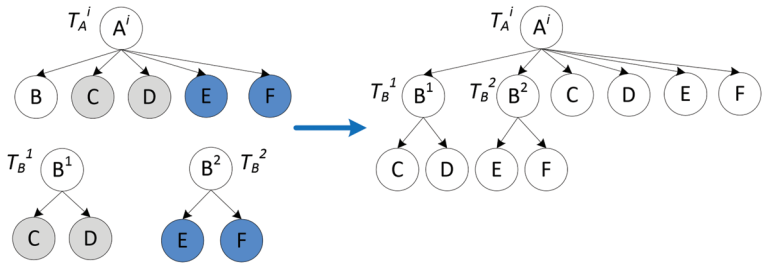


Figure 5. Vertical merge: multiple sense alignment.

3.3 Probability Calculation

Probability is an important feature of our taxonomy design. Different from previous approaches which treat knowledge as black and white, our solution is to live with noisy data with probability scores and let the application make the best use of it. We provide two kinds of probability scores, namely plausibility and typicality.

Plausibility is the joint probability of an isA pair. For each isA claim E, we use $p(\mathrm{E})$ to denote its probability to be true. Here we adopt a simple noisy-or model to calculate the probability. Assume E is derived from a set of sentences $\{s_1, s_2, \ldots, s_n\}$, a claim is false if every piece of evidence from $\{s_1, s_2, \ldots, s_n\}$ is false. Assuming every piece of evidence is independent, we have:

$$p(\mathrm{E}) = 1 - p(\bar{\mathrm{E}}) = 1 - \prod_{i=1}^{n}(1 - p(s_i)), \tag{5}$$

where $p(s_i)$ is the probability of evidence s_i to be true. We characterize each s_i by a set of features F_i including: (1) the PageRank score of the Web page from which sentence s_i is extracted; (2) the Hearst pattern used to extract evidence pair $<x, y>$ from s_i; (3) the number of sentences with x as the super-concept; (4) the number of sentences with y as the sub-concept; (5) the number of sub-concepts extracted from sentence s_i; (6) position of y in the sub-concepts list from sentence s_i. Supposing the features are independent, we apply Naïve Bayes [14] to derive $p(s_i)$ based on the corresponding feature set F_i. We exploit WordNet to build a training set used for learning the Naïve Bayes Model. Given a pair $<x, y>$ that appears in the WordNet, if there is a path between x and y in the WordNet taxonomy, (i.e., x is an ancestor of y), we regard the pair as a positive example; otherwise, we treat the pair as negative.

Typicality is the conditional probability between a super-concept and its instance (sub-concept). Intuitively, *robin* is more typical of the concept *bird* than is *ostrich*, while *Microsoft* is more typical of the concept *company* than is *Xyz inc*. Therefore, we need a probability score to stand for the typicality of an instance (sub-concept) to its corresponding super-concept. The typicality measure of an instance i to a super-concept x is formulated as:

$$\gamma(i|x) = \frac{n(x,i) \cdot p(x,i)}{\sum_{i' \in I_x} n(x,i') p(x,i')}, \tag{6}$$

where x is a super-concept, i is an instance of x, $n(x, i)$ is the number of appearance that supports i as an instance of x; $p(x, i)$ is the plausibility of pair $<x, i>$, and I_x is a set consisting of all instances of super-concept x. For example, suppose x = Company, i = Microsoft, j = Xyz inc. It can be imagined that $n(x, i)$ should be much larger than $n(x, j)$, so *Microsoft* should obtain a much bigger typicality score than *Xyz Inc* with respect to *Company*. Similarly, we can also define the typicality score denoting the probability of a concept x to instance i.

$$\gamma(x|i) = \frac{n(x,i) \cdot p(x,i)}{\sum_{x' \in I_i} n(x',i) p(x',i)}. \tag{7}$$

4. CONCEPTUALIZATION

In this section, we introduce the conceptualization model which leverages the Probase semantic knowledge network to facilitate text understanding. Conceptualization model (also known as the Concept Tagging model) aims to map text into semantic concept categories with some probabilities. It provides computers the common sense of semantics and makes machines "aware" of the mental world of human beings, through which the machines can better understand human communication in text.

We consider three sub-tasks for building a conceptualization model. (1) Single Instance Conceptualization, which returns Basic-Level Categorization (BLC) for a single instance. As an example, "Microsoft" could be automatically mapped to *Software Company* and *Fortune 500 company*, etc., with some prior probabilities. (2) Context-Aware Single Instance Conceptualization, which produces the most appropriate concepts based on different contexts. As an example, "Apple" could be mapped to *Fruit* or *Company* without context, but with context word "pie", "Apple" should be mapped to *Fruit* with higher probability. (3) Short Text Conceptualization, which returns the types and concepts related to a short sequence of text. For example, in the sentence "He is playing game on Apple iPhone and eating an apple", the first "Apple" is *Company* while the second "Apple" is *Fruit*.

4.1 Single Instance Conceptualization

Assume e is an instance, and c is a super-concept; we can obtain the Basic-Level Categorization (BLC) results based on typicality scores $P(e|c)$ and $P(c|e)$, where $P(e|c)$ denotes the typicality of an instance e to concept c, and $P(c|e)$ denotes the typicality of a concept c to instance e. We propose several metrics as representative measures used for BLC [15].

MI is the mutual information between e and c, defined as:

$$MI(e,c) = P(e,c) log \frac{P(e,c)}{P(e)P(c)}. \tag{8}$$

PMI denotes the pointwise mutual information, which is a widely used measure of the association between two discrete variables.

$$PMI(e,c) = log \frac{P(e,c)}{P(e)P(c)} = log \frac{P(e|c)P(c)}{P(e)P(c)} = log P(e|c) - log P(e). \tag{9}$$

NPMI is a normalized version of PMI, which is proposed to make PMI less sensitive to occurrence frequency and is more interpretable.

$$NPMI(e,c) = \frac{PMI(e,c)}{-logP(e,c)} = \frac{\log P(e|c) - logP(e)}{-logP(e,c)}. \tag{10}$$

Both PMI and NPMI suffer from the trade-off between general and specific concepts. On the one hand, general concepts may be the correct answer, but they do not have the capability to distinguish instances of different sub-categories; on the other hand, specific concepts may be more representative, but the coverage is low. Therefore, we further propose PMI^k and Graph Traversal measures to tackle this problem.

PMI^k makes a compromise to avoid producing either too general or too specific concepts.

$$Rep(e,c) = P(c|e)P(e|c). \tag{11}$$

If we take the logarithm of scoring function $Rep(e,c)$, we get:

$$\log Rep(e,c) = log\frac{P(e,c)^2}{P(e)P(c)} = PMI(e,c) + \log P(e,c). \tag{12}$$

This in fact corresponds to PMI^2, which is a normalized form in the PMI^k Family [16].

Graph Traversal is a common way to calculate the relatedness of two nodes in a large network. The scores calculated by general graph transversal can be formulated as:

$$\begin{aligned} Time(e,c) = \sum_{k=1}^{\infty}(2k) * P_k(e,c) = \sum_{k=1}^{T}(2k) * P_k(e,c) + \sum_{k=T+1}^{\infty}(2k) * P_k(e,c) \geq \sum_{k=1}^{T}(2k) * P_k(e,c) \\ +2(T+1) * \left(1 - \sum_{k=1}^{T} P_k(e,c)\right), \end{aligned} \tag{13}$$

where $P_k(e,c)$ is the probability of staring from e to c and back to e in $2k$ steps. When $k = 2$ which represents a 4-step random walk, we have:

$$Time'(e,c) = 2 * P(c|e)P(e|c) + 4 * (1 - P(c|e))P(e|c)) = 4 - 2 * P(c|e)P(e|c) = 4 - 2 * Rep(e,c). \tag{14}$$

Thus, it is verified that the simple, easy-to-compute scoring method of $Rep(e,c)$ is equivalent to a graph traversal approach under the constraint of 4-step random walk.

4.2 Context-Aware Single Instance Conceptualization

One instance may be mapped to distinct concepts according to different contexts. For example, for "apple ipad", we want to annotate "apple" with *company* or *brand*, and "ipad" with *device* or *product*. The major challenge is how to distinguish and detect the correct concepts in different contexts. We propose a framework of context-aware single instance conceptualization [17] which consists of two parts: (1) offline knowledge graph construction and (2) online concept inference.

4.2.1 Offline Knowledge Graph Construction

There are millions of concepts in the Probase semantic network. We first perform clustering on all the concepts to construct the concept clusters. Concretely, we adopt a *K*-Medoids clustering algorithm [18] to group the concepts into 5,000 clusters. We adopt concept clusters instead of concepts themselves in the graph for noise reduction and computation efficiency. In the rest of this section, we use concept to denote concept cluster. We build a large knowledge graph offline, which is comprised of three kinds of sub-graphs, including (1) instance to concept graph, (2) concept to concept graph and (3) non-instance term to concept graph. Figure 6 shows a piece of the graph around the term *watch*.

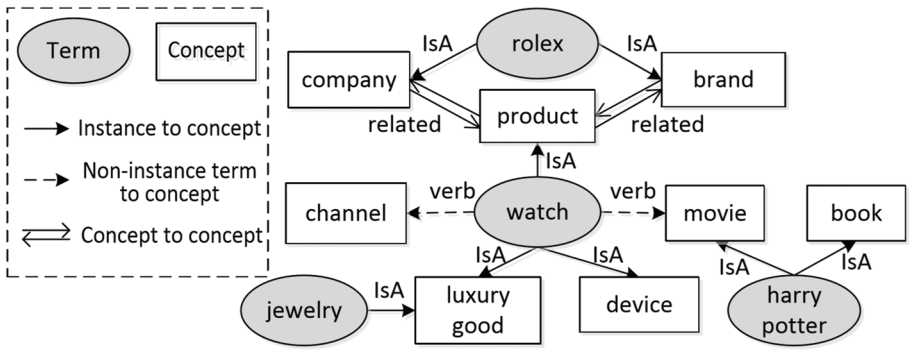


Figure 6. A subgraph of heterogeneous semantic network around *watch*.

Instance to concept graph. We directly use the Probase semantic network as the instance to concept graph. $P(c|e)$ is served as the typicality probability of concept (cluster) c to instance e, which can be computed by

$$P(c \mid e) = \sum_{c^* \in c} P^*(c^* \mid e), \tag{15}$$

where c^* represents a concept belonging to cluster c, $P^*(c^*|e)$ is the typicality of the concept c^* to instance e.

Concept to concept graph. We assign a transition probability $P(c_2|c_1)$ to the edge between two concepts c_1 and c_2. The probability is derived by aggregating the co-occurrences between all (unambiguous) instances of the two concepts.

$$P(c_2 \mid c_1) = \frac{\sum_{e_i \in c_1, e_j \in c_2} n(e_i, e_j)}{\sum_{c \in C} \sum_{e_i \in c_1, e_j \in c} n(e_i, e_j)}, \tag{16}$$

where c denotes a set containing all concepts (clusters), and the denominator is applied for normalization. In practice, we only take the top 25 related concepts for each concept for edge construction.

Non-instance term to concept graph. There are terms that cannot be matched to any instances or concepts, i.e., verbs or adjectives. For better understanding of the short text, we also mine lexical relationships

between non-instance terms and concepts. Specifically, we use the Stanford Parser [19] to obtain POS tagging and dependency relationships between tokens in the text, and the POS tagging results reveal the type (e.g., adjective, verb, etc.) of each token. Our goal is to obtain the following two distributions.

$\boldsymbol{P(z|t)}$ stands for the prior probability that a term t is of a particular type z. For instance, the word *watch* is a verb with probability $P(verb|watch) = 0.8374$. We compute the probability as $P(z|t) = \frac{n(t,z)}{n(t)}$, where $n(t, z)$ is the frequency term t annotated as type z in the corpus, and $n(t)$ is the total frequency of term t.

$P(c|t, z)$ denotes the probability of a concept c, given the term t of a specific type z. For example, $P(movie|watch, verb)$ depicts how likely the verb *watch* is associated with the concept *movie* lexically. Specifically, we detect co-occurrence relationships between instances, verbs and adjectives in Web documents parsed by the Stanford Parser. To obtain meaningful co-occurrence relationships, we require that the co-occurrence be embodied by dependency, rather than merely appearing in the same sentence. We first obtain $P(e|t, z)$, which denotes how likely a term t of type z co-occurs with instance e:

$$P(e|t,z) = \frac{n_z(e,t)}{\sum_{e^*} n_z(e^*,t)}, \tag{17}$$

where $n_z(e, t)$ is the frequency of term t and instance e having a dependency relationship when t is of type z. Then, taking instances as the bridge, we calculate the relatedness between non-instance terms (adjectives/verbs) and concepts.

$$P(c|t,z) = \sum_{e \in c} P(c,e|t,z) = \sum_{e \in c} P(c|e) \times P(e|t,z). \tag{18}$$

In Equation (18), $e \in c$ means that e is an instance of concept c, and we make an assumption that a concept c is conditionally independent with term t and type z when the instance e is given.

4.2.2 Online Concept Inference

We adopt the heterogeneous semantic graph built offline to annotate the concepts for a query. First, we segment the query into a set of candidate terms which use Probase as lexicon and identify all occurrences of terms in the query. Second, we retrieve the subgraph out of the heterogeneous semantic graph by concentrating on the query terms. Finally, we perform multiple random walks on the sub-graph to find the concepts with the highest weights after convergence.

4.3 Short Text Conceptualization

Short text is hard to understand in three aspects: (1) Compared with long sentences, short sentences lack syntactic features and cannot directly apply POS tagging; (2) Short texts do not have sufficient statistical signals; (3) Short text is usually ambiguous due to the lack of context terms. Many research works focus on statistical approaches like topic models [20], which extract latent topics as implicit semantics. However, we argue that semantic knowledge is needed to get a better understanding of short texts. Thus, we aim to

utilize the semantic knowledge network to enhance short text understanding [21]. Different from the knowledge graph built for context-aware single instance conceptualization (Section 4.2), we add a novel sub-graph, typed-term co-occurrence graph, which is an undirected graph where nodes are typed-terms and edge weights represent the lexical co-occurrence between two typed-terms. Nevertheless, the number of typed-terms is extremely large, which will increase storage cost and affect the efficiency of calculation on the network. Therefore, we compress the original typed-term co-occurrence network by retrieving Probase concepts for each instance. Then, the typed-terms can be replaced by the related concept clusters and the edge weights are aggregated accordingly. Figure 7 shows an example of the compression procedure.

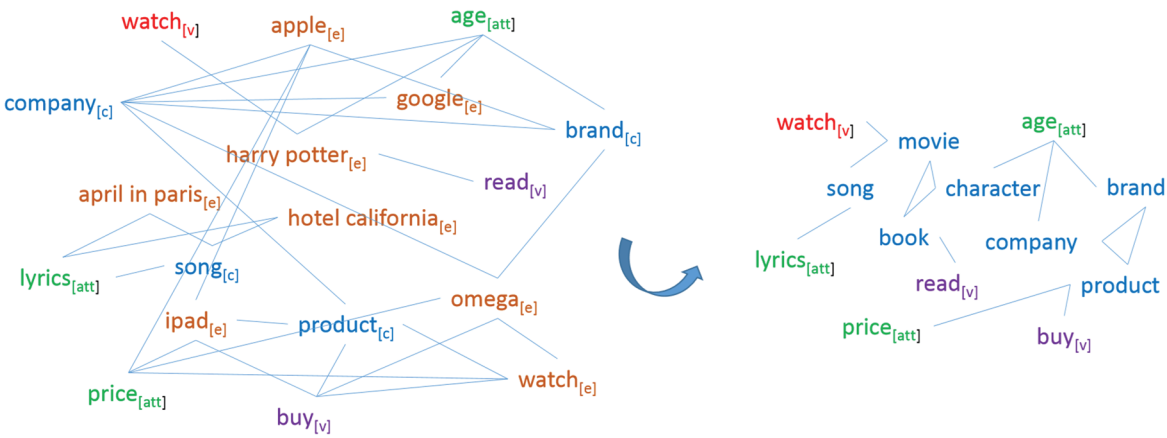


Figure 7. The compression procedure of typed-term co-occurrence network.

Given a piece of short text, each term is associated with the type and corresponding concepts. We define the types as *instance*, *verb*, *adjective* and *concept*. For each term with type *instance*, we also learn the corresponding concepts for the term. Given a piece of short text, the online inference procedure contains three steps, including text segmentation, type detection and concept labeling. An example is illustrated in Figure 8. Given the query "book disneyland hotel california", the algorithm first segments it as "book disneyland hotel california"; then, it detects the type for each segment; at last, the concepts are labeled for each instance. As shown in the figure, the output is "book$_{[v]}$ disneyland$_{[e](park)}$ hotel$_{[c]}$ california$_{[e](state)}$", which means that book is a *verb*, disneyland is an *instance* of the concept *park*, hotel is a *concept*, and california is an *instance* of the concept *state*.

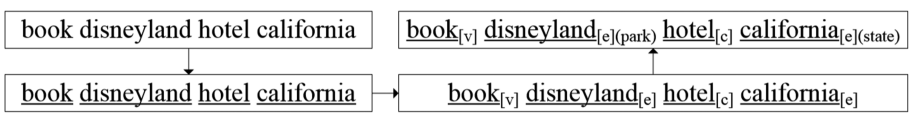


Figure 8. An example of short text understanding.

4.3.1 Text Segmentation

The goal of text segmentation is to divide a short text into a sequence of meaningful components. To determine a valid segmentation, we define two heuristic rules: (1) except for stop words, each word belongs to one and only one term; (2) terms are coherent (i.e., terms mutually reinforce each other). Intuitively, {april in paris lyrics} is a better segmentation of "april in paris lyrics" than {april in paris lyrics}, since "lyrics" is more semantically related to *songs* than *months* or *cities*. Similarly, {vacation april in paris} is a better segmentation of "vacation april in paris" than {vacation april in paris}, because "vacation", "april" and "paris" are highly coherent with each other, while "vacation" and "april in paris" are less coherent.

The text segmentation algorithm can be conducted in the following steps. First, we construct a term graph (TG) which consists of candidate terms and their relationships. Next, we add an edge between two candidate terms when they are not mutually exclusive and set the edge weight to reflect the strength of mutual reinforcement. Finally, the problem of finding the best segmentation is transformed into the task of finding a clique in the original TG, with 100% word coverage while maximizing the average edge weights.

4.3.2 Type Detection

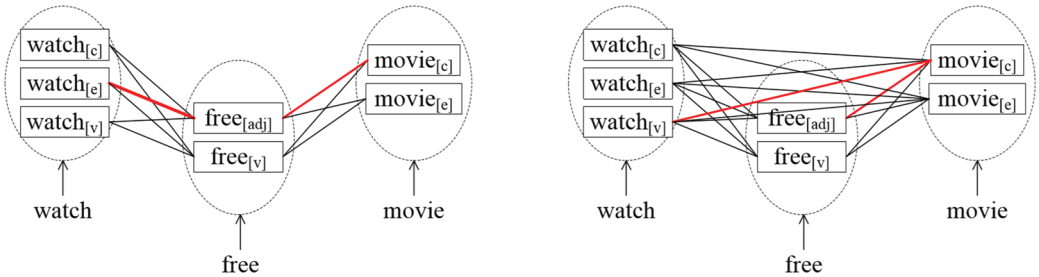
The type detection procedure annotates the type for each term as *verb*, *adjective*, *instance* or *concept*. For example, term "watch" appears in the instance-list, concept-list, as well as verb-list of our vocabulary, and thus the possible typed-terms of "watch" are $\text{watch}_{[c]}$, $\text{watch}_{[e]}$ and $\text{watch}_{[v]}$. For each term, the type detection algorithm determines the best typed-term from the set of possible candidates. In the case of "watch free movie", the best typed-terms for "watch", "free", and "movie" are $\text{watch}_{[v]}$, $\text{free}_{[adj]}$ and $\text{movie}_{[c]}$, respectively.

Traditional approaches resort to POS tagging algorithms which consider lexical features only, e.g., Markov Model [22]. However, such surface features are insufficient to determine types of terms especially in the case of short text. In our solution, we calculate the probability by considering both traditional lexical features and semantic coherence features. We formulate the problem of type detection into a graph model and propose two models, namely Chain model and Pairwise model.

Chain model borrows the idea of first order bilexical grammar and considers topical coherence between adjacent typed-terms. In particular, we build a chain-like graph where nodes are typed-terms retrieved from the original short text. Edges are added between each pair of typed-terms mapped from adjacent terms in the original text sequence, and the edge weights between typed-terms are calculated by affinity scores (see the example in Figure 9(a)). Chain model only considers semantic relations between consecutive terms which will lead to mistakes.

Pairwise model adds edges between typed-terms mapped from each pair of terms rather than adjacent terms only. As an example, in Figure 9 (b), there are edges between non-adjacent terms "watch" and "movie". The goal of the Pairwise Model is to find the best sequence of typed-terms which guarantees that the maximum spanning tree (MST) constructed by the selected sequence has the largest total weight. As

shown in Figure 9 (b), as long as the total weight of edges between ($watch_{[v]}$, $movie_{[c]}$) and ($free_{[adj]}$, $movie_{[c]}$) is the largest, {$watch_{[v]}$, $free_{[adj]}$, $movie_{[c]}$} can be successfully recognized as the best sequence of type detection for the query "watch free movie". We employ the Pairwise model in our system as it achieves higher accuracy experimentally compared with the Chain model.



(a) Type detection result of "watch free movie using the Chain model is {$watch_{[e]}$, $free_{[adj]}$, $movie_{[c]}$}".

(b) Type detection result of "watch free movie using the Pairwise model is {$watch_{[v]}$, $free_{[adj]}$, $movie_{[c]}$}".

Figure 9. Example of Chain model and Pairwise model.

4.3.3 Concept Labeling

The goal of concept labeling is to re-rank the candidate concepts according to the context for each instance. The most challenging task in concept labeling is to deal with ambiguous instances. Our intuition is that a concept is appropriate for an instance only if it is a common semantic concept of that instance and is supported by the surrounding context at the same time. Take "hotel California eagles" as an example, where both *animal* and *music band* are popular semantic concepts to describe "eagles". If we find a concept *song* in the context, we can be more confident that *music band* should be the correct concept for "eagles".

After type detection, we have obtained a set of instances for concept labeling. For each instance, we collect a set of concept candidates and perform instance disambiguation based on a Weighted-Vote approach, which is a combination of Self-Vote and Context-Vote. Self-Vote denotes the original affinity weight (calculated by normalized co-occurrence) of a concept cluster *c* associated to the target instance; while Context-Vote leverages the affinity weights between the target instance and other concepts found in the context. In the case of "hotel california eagles", the original concept vector of the instance *eagles* is (<animal, 0.2379>, <band, 0.1277>, <bird, 0.1101>, <celebrity, 0.0463>, …) and the concept vector for context "hotel california" is (<singer, 0.0237>, <band, 0.0181>, <celebrity, 0.0137>, <album, 0.0132>, …). After disambiguation by Weighted-Vote, the final conceptualization result of *eagles* is (<band, 0.4562>, <celebrity, 0.1583>, <animal, 0.1317>, …).

5. APPLICATION

Probase Browser shows the backbone of the Probase taxonomy and provides an interface to search for super-concepts, sub-concepts, as well as similar concepts corresponding to the given query. Figure 10 is a snapshot of the Probase browser.

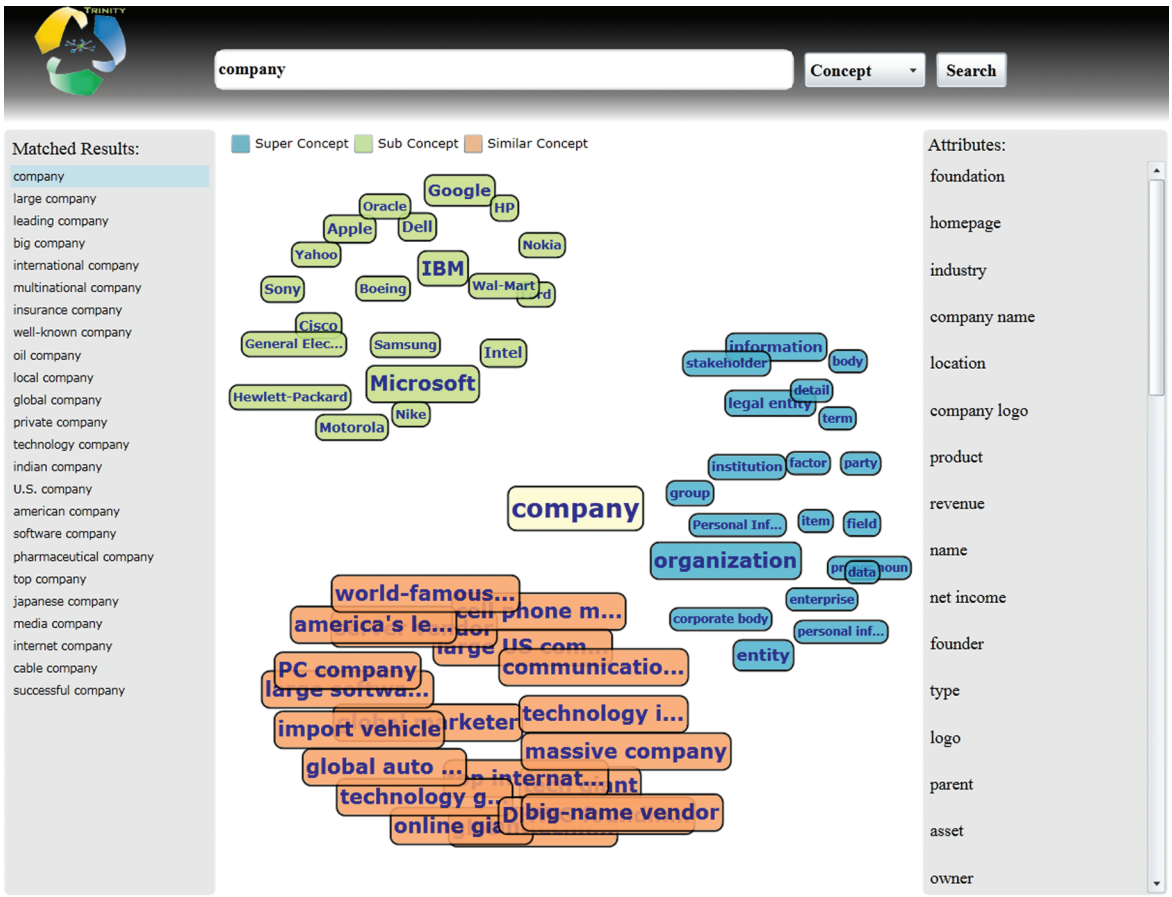


Figure 10. A snapshot of the Probase browser.

Tagging Service provides the capability of tagging a piece of text with a concept vector, based on which the semantic similarity can be calculated, and various text processing applications can be affiliated. Figure 11 shows a snapshot of the instance conceptualization demo when querying a single instance "python". Figure 12 shows the snapshot given "apple" in the context "pie" and "ipad", respectively. As shown in the figure, our tagging service can map the term "apple" into correct concepts under different contexts. An example of short text conceptualization is illustrated in Figure 13 by querying the tagging service with "apple engineer is eating the apple". The result shows the capability of our tagging algorithm to distinguish different senses of "apple" in the short text scenario.

Score by P(c|e)		Score by MI		Score by P(e|c)		Score by NPMI		Score by PMI^K		Score by BLC	
language	0.505	language	0.452	living snake	0.1	dynamic language	0.107	dynamic language	0.143	dynamic language	0.245
scripting language	0.11	scripting language	0.135	noticed danger	0.1	large snake	0.105	large snake	0.132	scripting language	0.192
programming language	0.101	programming language	0.108	actual snake	0.1	scripting language	0.102	exotic leather product	0.11	large snake	0.151
dynamic language	0.073	dynamic language	0.1	non-native snake	0.1	exotic leather product	0.1	living snake	0.097	language	0.121
snake	0.071	snake	0.077	decent programming	0.1	living snake	0.098	noticed danger	0.097	programming language	0.072
animal	0.036	large snake	0.046	interprated language	0.1	noticed danger	0.098	actual snake	0.097	snake	0.054
reptile	0.035	reptile	0.028	fully-fledged programming	0.1	actual snake	0.098	scripting language	0.092	exotic leather product	0.047
large snake	0.033	exotic skin	0.024	modern high-level scripting language	0.1	dynamic scripting language	0.098	dynamic scripting language	0.089	interpreted language	0.044
exotic skin	0.02	interpreted language	0.019	high-order programming	0.1	primitive snake	0.096	nocturnal snake	0.071	primitive snake	0.04
technology	0.016	primitive snake	0.012	biggest snake	0.1	dynamically typed language	0.096	primitive snake	0.071	dynamically typed language	0.034

Figure 11. Snapshot of single instance conceptualization.

pie		apple	
[3/product]		[9405/food]	
3/product	**0.2643321**	**9405/food**	**0.4323602**
baked good	0.01293272	food	0.2664687
product	0.0105567	sweet food	0.03745969
bakery product	0.008165566	sugary food	0.02410059
meat product	0.006891184	snack food	0.02355857
baked product	0.005766741	rich food	0.01787281
processed meat product	0.004463319	raw food	0.01127649
homemade baked good	0.004463319	staple food	0.01009054
commercially baked good	0.004463319	comfort food	0.006366423
home-baked good	0.004463319	carbohydrate food	0.006366423
bake good	0.004463319	starchy food	0.006366422
9405/food	**0.252832**	**196/dessert**	**0.1742629**
food	0.01338996	dessert	0.09348933
fatty food	0.007688988	goodie	0.03226487
snack food	0.007113159	treat	0.02879891
processed food	0.005766741	fruit dessert	0.009326639
sugary food	0.005766741	homemade dessert	0.006366422
sweet food	0.00539643	fruit-based dessert	0.004016765
rich food	0.00539643		
hot food	0.004968937		
prepared food	0.004463319		
dessert food	0.004463319		

ipad		apple	
[15/device]		[1/company]	
15/device	**0.4352382**	**1/company**	**0.926092**
device	0.01287828	company	0.01018514
mobile device	0.01198808	corporation	0.00609147
portable device	0.009427363	firm	0.005942625
apple device	0.008862641	large company	0.005684517
tablet device	0.00882206	client	0.005453772
ios device	0.008738589	player	0.005367511
gadget	0.00836913	stock	0.005275559
electronic device	0.007205624	technology company	0.005275559
handheld device	0.005836552	big company	0.004995803
digital device	0.005672655	giant	0.0048316
3/product	**0.09435893**	**1053/top brand name/brand**	**0.0393841**
product	0.009799219	brand	0.001478651
apple product	0.009299118	popular brand	0.000855937
electronic product	0.003065936	name brand	0.000782961
apple's product	0.003065936	big brand	0.000720103
popular product	0.002429697	great brand	0.000701757
digital product	0.002429697	global brand	0.000701757
revolutionary product	0.002429697	top brand	0.000660766
iconic product	0.002429697	well-known brand	0.000637629
popular apple product	0.002429697	iconic brand	0.000637629
apple's high technology product	0.002429697	laptop brand	0.000612285

Figure 12. Snapshot of context-aware single instance conceptualization.

ShortText: apple engineer is eating the apple Conceptualize

apple		engineer		eat	apple	
[1/company]		[805/professional]		[verb]	[9405/food]	
1/company	**0.9481527**	**805/professional**	**0.3667608**		**9405/food**	**0.9647822**
company	0.0104278	professional	0.01444558		food	0.01994285
corporation	0.006236602	expert	0.008747877		ingredient	0.01210647
firm	0.00608421	occupation	0.008747877		high fiber food	0.0108261
large company	0.005819953	design professional	0.007727818		hard food	0.01037435
client	0.00558371	licensed professional	0.006690023		crunchy food	0.009956987
player	0.005495394	technical	0.006299564		fiber-rich food	0.009842971
stock	0.005401252	professional group	0.00599617		healthy food	0.009724479
technology company	0.005401252	skilled professional	0.00599617		fresh food	0.009338287
big company	0.00511483	construction	0.005645925		fiber rich food	0.008181235
giant	0.004946716	industry professional	0.004724673		wholesome	0.007972804
9405/food	**0.02624887**	**355/staff/job**	**0.3131405**		**3/product**	**0.02158811**
food	0.000542585	job	0.009024879		product	0.001138903
ingredient	0.00032938	skilled worker	0.008241975		farm product	0.000464368
high fiber food	0.000294545	knowledge worker	0.007390991		private good	0.000464368
hard food	0.000282255	technical staff	0.00728621		local product	0.000417116
crunchy food	0.000270899	worker	0.006940636		company's product	0.000417116
fiber-rich food	0.000267797	professional worker	0.005652321		branded product	0.000359284
healthy food	0.000264574	staff	0.0051793		seasonal product	0.000359284
fresh food	0.000254066	white-collar worker	0.0051793		bulk product	0.000359284
fiber rich food	0.000222587	professional	0.004901662		well-known product	0.000359284
wholesome	0.000216916	nonproduction	0.004586902		horticultural product	0.000359284

Figure 13. An example of short text conceptualization.

Topic Search [23] aims at understanding the topic underlined in each query for better search relevance. Figure 14 shows a snapshot when a user queries "database conference in Asian cities". As shown in the figure, the search results correctly rank VLDB 2002, 2006, and 2010 at the top, which are held in Hong Kong, Seoul and Singapore, respectively. Traditional Web search takes queries as sequences of keywords instead of understanding the semantic meanings, so it is hard to generate the correct answers for the example query. To achieve this goal, we present a framework that improves Web search experiences using Probase knowledge and the conceptualization models. First, it classifies Web queries into different patterns according to the concepts and entities in addition to keywords contained in the queries. Then, it produces answers by interpreting the queries with the help of Probase semantic concepts. Our preliminary results showed that the framework was able to understand various types of topic-like queries and achieved much higher user satisfaction.

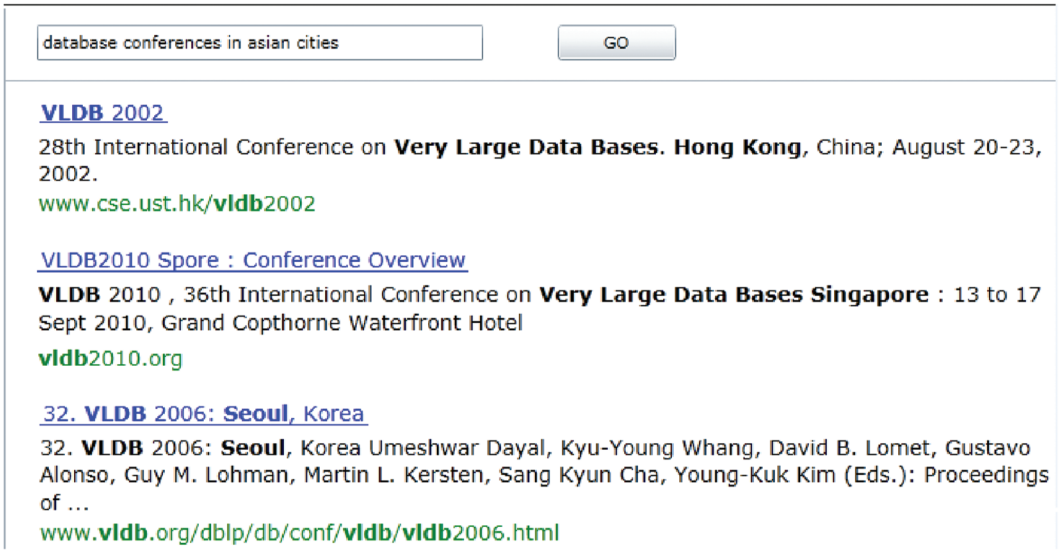


Figure 14. The framework of topic search.

Understanding Web Tables [24]. We use Probase to help interpret and understand Web tables, which unlocks the wealth of information hidden in the Web pages. To tackle this problem, we build a pipeline for detecting Web tables, understanding their contents, and applying the derived knowledge to support semantic search. From 300 million Web documents, we extract 1.95 billion raw HTML tables. Many of them do not contain useful or relational information (e.g., used for page layout purpose); others have structures that are too complicated for machines to understand. We use a rule-based filtering method to acquire 65.5 million tables (3.4% of all the raw tables) that contain potentially useful information. We adopt Probase taxonomy to facilitate the understanding of table content, by associating the table content with one or more semantic concepts in Probase. Based on the knowledge mined from Web tables, we build a semantic search engine over tables to demonstrate how structured data can empower information retrieval on the Web. A snapshot of our semantic search engine is shown in Figure 15. As illustrated in the figure, when a user queries "American politicians birthday", the search engine returns with aggregated Web tables consisting of birthday and other related knowledge of various American politicians.

American politicians birthday

Web Table More▾

Shrink

Shrink table

Birth order	U.S. Vice President	Birthdate	Century	Order of office	Birthplace
39	Richard Nixon	January 9, 1913	20th	36	Yorba Linda , California
28	Theodore Roosevelt	October 27, 1858	19th	25	New York City , New York
46	Dan Quayle	February 4, 1947	20th	44	Indianapolis , Indiana
38	Hubert Humphrey	May 27, 1911	20th	38	Wallace , South Dakota
40	Gerald Ford	July 14, 1913	20th	40	Omaha , Nebraska
42	George H. W. Bush	June 12, 1924	20th	43	Milton , Massachusetts
44	Dick Cheney	January 30, 1941	20th	46	Lincoln , Nebraska
45	Joseph Biden	November 20, 1942	20th	47	Scranton , Pennsylvania
9	Martin Van Buren	December 5, 1782	18th	8	Kinderhook , New York

Shrink table

State	Senator	Party	Date of birth	Term	Age (Years/Days)
Illinois	Barack Obama	Democratic	August 4, 1961	2005 - 2008	1961 8 4
New York	Hillary Clinton	Democratic	October 26, 1947	2001 - 2009	1947 10 26
Tennessee	Al Gore	Democratic	March 31, 1948	1985 - 1993	1948 3 31
North Carolina	John Edwards	Democratic	June 10, 1953	1999 - 2005	1953 6 10
Kansas	Bob Dole	Republican	July 22, 1923	1969 - 1996	1923 7 22
Indiana	Dan Quayle	Republican	February 4, 1947	1981 - 1989	1947 2 4

Figure 15. Snapshot of the Web tables.

Channel-based Query Recommendation [25] aims to anticipate user search needs when browsing different channels, by recommending the hottest and highly related queries for a given channel. As shown in Figure 16, there are three channels, News, Sports and Entertainment, and several queries are recommended under each channel to enable the users to explore the hottest topics related to the target channel. One of the main challenges is how to represent queries and channels which are short pieces of text. Traditional

representation methodology treats text as bag of words, which suffers from mismatch of surface forms of words, especially when text is short. Therefore, we leverage the Probase taxonomy to obtain a "Bag of Concepts" representation for each query and channel, in order to avoid surface mismatching and handle the problem of synonym and polysemy. By leveraging the large taxonomy knowledge base of Probase, we learn a concept model for each category, and conceptualize a short text to a set of relevant concepts. Moreover, a concept-based similarity mechanism is presented to classify the given short text to the most similar category. Experiments showed that our framework could map queries to channels with a high degree of precision (average precision = 90.3%).

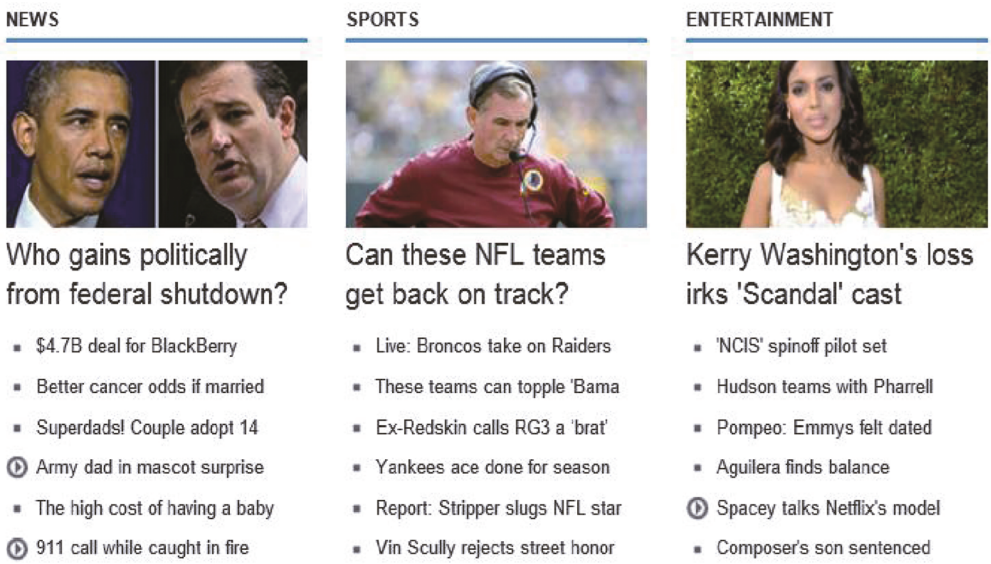


Figure 16. Query recommendation snapshot.

Ads Relevance [3]. In sponsored search, the search engine maps each query to the related ad bidding keywords. Since both the query and bidding keywords are short, the traditional bag-of-words approaches do not work well in this scenario. Therefore, we can leverage Probase concept taxonomy to enhance Ads relevance calculation. For each short text, we first identify instances from it, and map it to basic-level concepts with score *Rep*(*e*, *c*); then, we merge the concepts to generate a concept vector representing the semantics of the target text. Finally, the relevance score can be calculated through the cosine similarity measure between the concept vectors of the query and bidding keywords. We conduct our experiments on real Ads click logs collected from Microsoft Bing search engine. We calculate the relevance score of each candidate pair of query and bidding keywords, divide the pairs into 10 buckets based on the relevance score, and report the average clickthrough rate (CTR) within each bucket. The result is demonstrated in Figure 17, which shows that the CTR numbers have a strong linear correlation with the relevance scores calculated by our model.

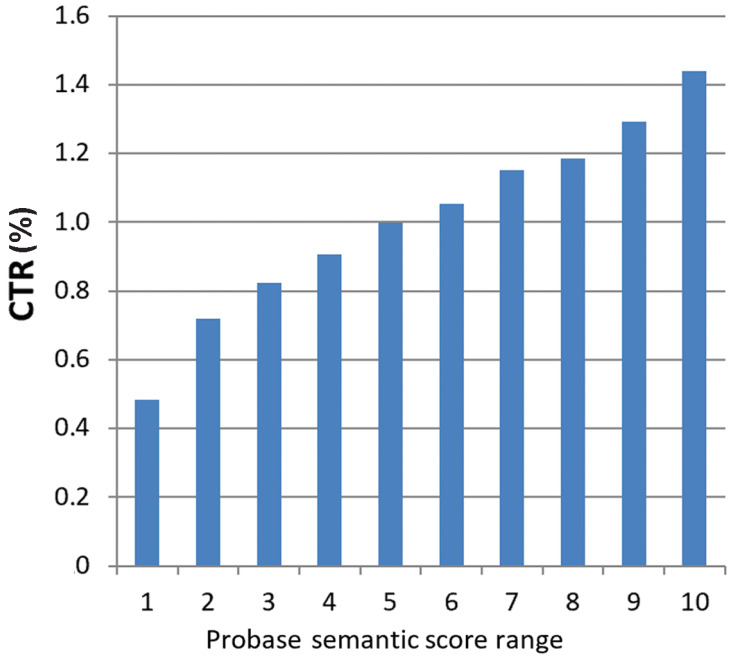


Figure 17. The correlation of CTR with Ads relevance score.

5. DATA AND ANALYSIS

5.1 Data

The Microsoft Concept Graph can be downloaded at https://concept.research.microsoft.com/, which is a sub-graph of the semantic network we introduce in this paper. The core taxonomy of Microsoft Concept Graph contains above 5.4 million concepts, whose distribution is shown in Figure 18, where *Y* axis is the number of instances each concept contains (logarithmic scale), and on the *X* axis are the 5.4 million concepts ordered by their size. Our concept space is much larger than other existing knowledge bases (Freebase contains no more than 2,000 concepts, and Cyc has about 120,000 concepts).

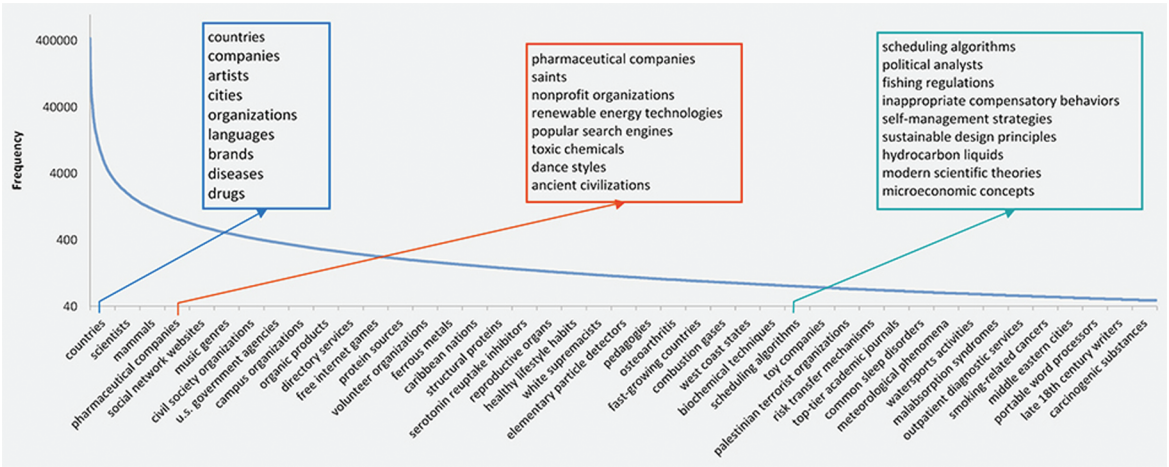


Figure 18. The distribution of concepts in Microsoft Concept Graph.

5.2 Concept Coverage

Based on the query logs of Microsoft Bing search engine, we estimate the coverage of concepts mined by our methodology. If at least one concept can be found in a query, the query is considered to be covered. We thus calculate the percentage of queries that can be covered by Probase and compare the metrics against the other four taxonomies. We utilize the top 50 million queries in Bing's query log to compute the coverage metrics, and the results are shown in Figure 19. We can see clearly from the figure that Probase has the largest coverage, YAGO ranks the second, and Freebase has a rather low coverage although its instance space is large. It demonstrates that Freebase's instance distribution is very skew (most instances are distributed in several top categories and lack the general coverage of other concepts).

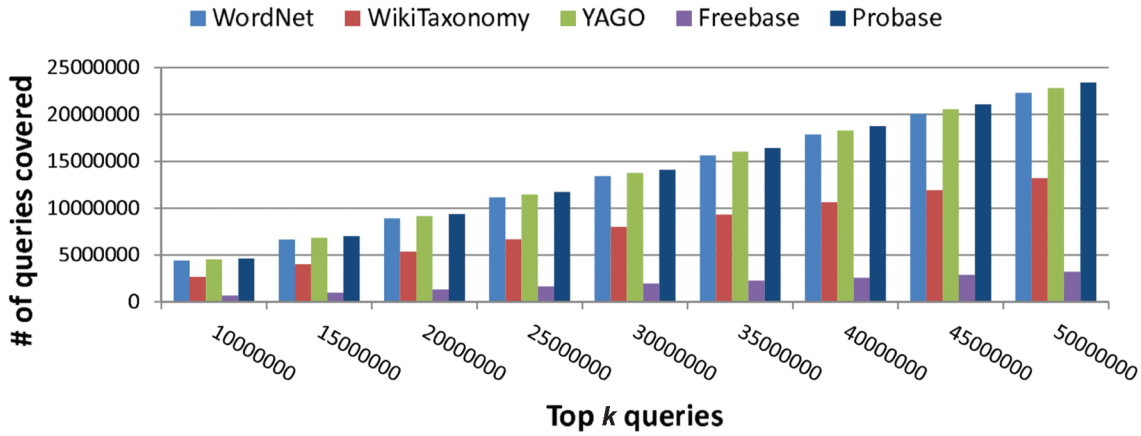


Figure 19. Concept coverage of different taxonomies.

5.3 Precision of isA pairs

To estimate the correctness of extracted isA pairs in Probase, we create a benchmark data set of 40 concepts in various domains. For each concept, we randomly pick up 50 instances (sub-concepts) and ask human reviewers to evaluate their correctness. Figure 20 depicts the precision of isA pairs for all the 40 concepts we manually evaluate. The overall precision is 92.8% by averaging over all the concepts.

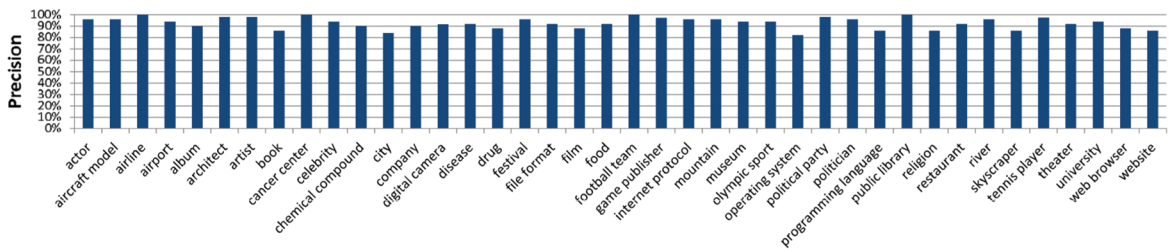


Figure 20. Precision of extracted isA pairs on 40 concepts.

We further draw the curve of average precision after each iteration (shown in Figure 21). Moreover, the number of concepts and isA pairs discovered after each iteration are drawn in Figure 22. It is obvious that the precision degrades after each round while the number of discovered concepts and isA pairs increases. So, the best iteration number must be chosen as a trade-off between precision and facts number (set as 11 in our implementation).

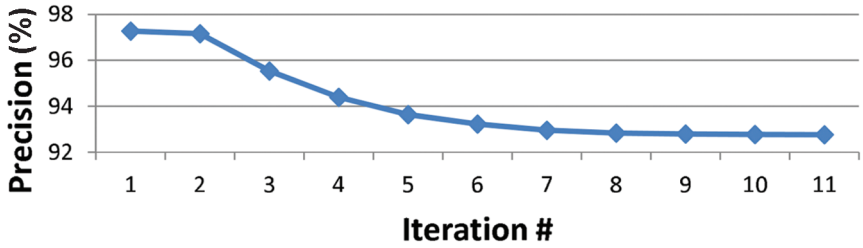


Figure 21. Precision of isA pairs after each iteration.

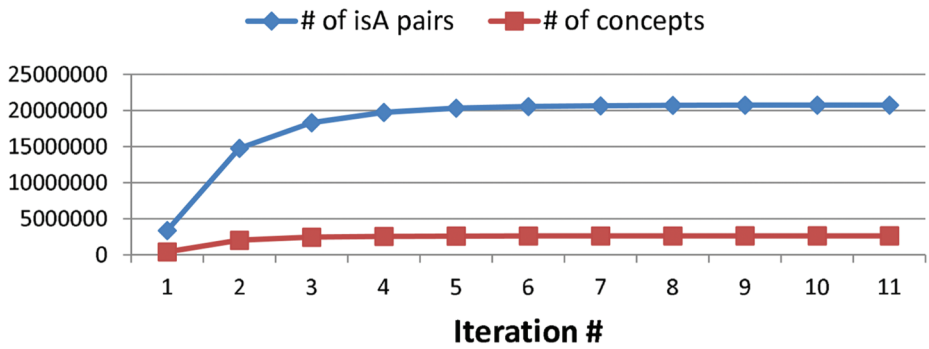


Figure 22. The number of discovered concepts and isA pairs after each iteration.

5.4 Conceptualization Experiment

In this section, we mainly present the experimental results of conceptualization for both single instance and short text.

1). Single Instance Conceptualization

Data set preparation. We asked human labelers to manually annotate the correctness of the concepts. The label is defined in four categories as shown in Table 3. Each (instance, concept) is assigned to three labelers to annotate. The final label is defined by the majority label. We will ask the fourth annotator to make the final vote for records without majority label. In all, there are 5,049 labeled records.

Table 3. Labeling guideline for conceptualization.

Label	Meaning	Examples
Excellent	Well matched concepts at the basic level	(bluetooth, wireless communication protocol)
Good	A little general or specific	(bluetooth, accessory)
Fair	Too general or specific	(bluetooth, feature)
Bad	Non-sense concepts	(bluetooth, issue)

Measurement. We employ Precision@k and nDCG to evaluate the effectiveness. For Precision@k, we treat Good/Excellent as score 1, and Bad/Fair as 0. We calculate the precision of top-k concepts as follows:

$$\text{Precision@}k = \frac{\sum_{i=1}^{k} rel_i}{k}, \tag{19}$$

where rel_i is the score we define above. For nDCG, we treat Bad as 0, Fair as 1, Good as 2, and Excellent as 3. Then we calculate nDCG as follows:

$$\text{nDCG} = \frac{rel_i + \sum_{i=2}^{k} \frac{rel_i}{logi}}{ideal\_rel_i + \sum_{i=2}^{k} \frac{ideal\_rel_i}{logi}}, \tag{20}$$

where rel_i is the relevance score of the result at rank i, and ideal rel_i is the relevance score at rank i of an ideal list, obtained by sorting all relevant concepts in decreasing order of the relevance score.

Experimental result. Figure 23 shows the evaluation of the top 20 results using both Precision and nDCG with and without smoothing. We compare our proposed scoring functions with various baselines: MI, NPMI, and PMI^3. Where PMI^3 is defined as:

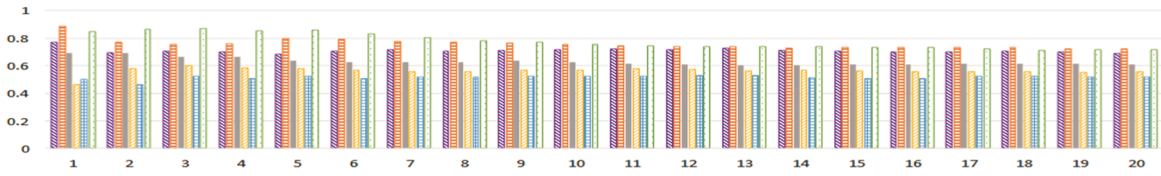
$$\text{PMI}^3(e,c) = log\frac{p(e,c)^3}{p(e)p(c)}. \tag{21}$$

From the result, we can see that our proposed scoring function outperforms baseline on both precision and nDCG.

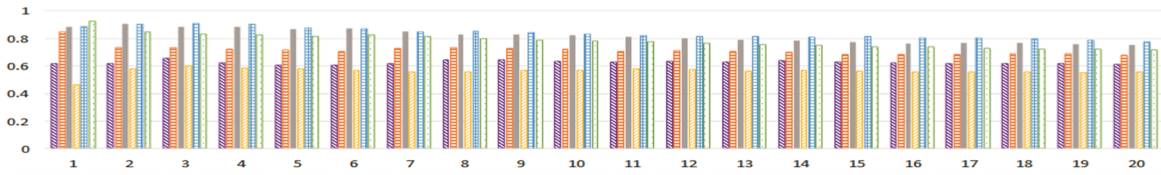
2). Short Text Conceptualization

Data set preparation. To validate our generalizability, we build two data sets to evaluate our algorithm including user search query and tweets. To build a query data set, we first manually picked 11 ambiguous terms including "apple", "fox" with instance ambiguity, "watch", "book", "pink", "blue", "population", "birthday" with type ambiguity, and "April in Paris", "hotel California" with segmentation ambiguity. Then we randomly selected 1,100 queries containing 11 ambiguous terms. Moreover, we randomly sampled another 400 queries without any restriction. In all, there are 1,500 queries. To build tweets data set, we also randomly sampled 1,500 tweets using Twitter's API. To clean the tweets, we removed some tweet-specific features, such as @username, hashtags, urls, etc. We asked human labelers to manually annotate the correctness. For each record, we assign at lease three labelers and pick up the majority vote as final label.

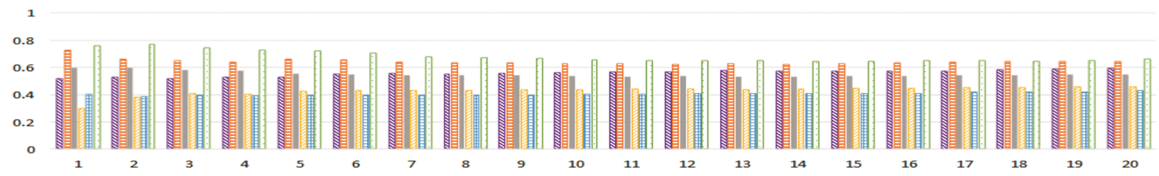
Experimental result. To evaluate the effectiveness of the algorithm, we compared our methods with several baseline methods. [27] conducts instance disambiguation in queries based on similar instances, [28] conducts instance disambiguation in queries based on related instances, and [26] conducts instance disambiguation in tweets based on both similar and related instances. Table 4 presents the results which show that our conceptualization is not only effective but also robust.



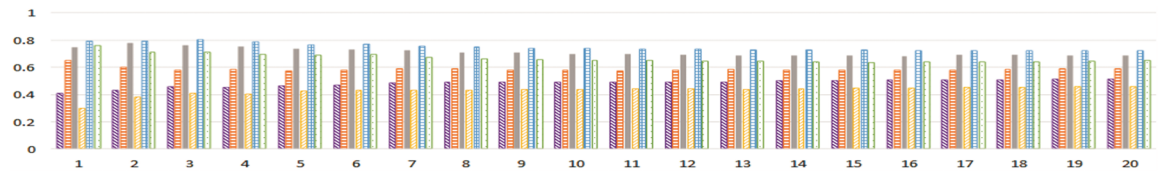
(a) Precision without smoothing



(b) Precision with smoothing ($\varepsilon = 1e\text{-}4$)



(c) nDCG without smoothing



(d) nDCG with smoothing ($\varepsilon = 1e\text{-}4$)

Figure 23. Precision and nDCG comparison.

Table 4. Precision of short text understanding.

	[27]	[28]	[26]	Ours
Query	0.694	0.701	-	0.943
Tweet	-	-	0.841	0.922

6. CONCLUSION

In this paper, we present the end-to-end framework of building and utilizing the Microsoft Concept Graph, which consists of three major layers, namely semantic network construction, concept conceptualization and applications. In the semantic network construction layer, we focus on the knowledge extraction and taxonomy construction procedures to build an open-domain and probabilistic taxonomy known as Probase. Like human mental process, we then represent text by concept space using conceptualization models, and empower many applications including topic search, query recommendation, Ads relevance calculation as well as Web table understanding. The system has received wide public attention ever since released in 2016 (more than 100,000 pageviews, 2 million API calls and 3,000 registered downloads from 50,000 visitors over 64 countries).

AUTHOR CONTRIBUTIONS

All of the authors contributed equally to the work. J. Yan (jun.yan@yiducloud.cn), Z. Wang (wzhy@outlook.com) and D. Zhang (zhangdawei@outlook.com) used to be the leaders of the Microsoft Concept Graph system, and now L. Ji (leiji@microsoft.com, corresponding author) is the leader and is developing new component for V2.0. Y. Wang (Yujing.Wang@microsoft.com) and L. Ji summarized the methodology part of this paper. Y. Wang and B. Shi (botianshi@bit.edu.cn) summarized the applications and experiment. All authors have made meaningful and valuable contributions in revising and proofreading the manuscript.

REFERENCES

[1] G. Murphy. The big book of concepts. Cambridge, MA: The MIT Press, 2004. isbn: 9780262632997.
[2] P. Bloom. Glue for the mental world. Nature 421 (2003), 212–213. doi: 10.1038/421212a.
[3] W. Wu, H. Li, H. Wang, & K. Zhu. Probase: A probabilistic taxonomy for text understanding. In: Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data, ACM, 2012, pp. 217–228. doi: 10.1145/2213836.2213891.
[4] D.B. Lenat, & R.V. Guha. Building large knowledge-based systems: Representation and inference in the Cyc project. Reading, MA: Addison-Wesley, 1990. isbn: 9780201517521.
[5] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, & J. Taylor. Freebase: A collaboratively created graph database for structuring human knowledge. In: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, ACM, 2008, pp. 1247–1250. doi: 10.1145/1376616.1376746.
[6] F.M. Suchanek, G. Kasneci, & G. Weikum. YAGO: A core of semantic knowledge. In: Proceedings of the 16th International Conference on World Wide Web, ACM, 2007, pp. 697–706. doi: 10.1145/1242572.1242667.
[7] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E.R.H. Jr., & T.M. Mitchell. Toward an architecture for never-ending language learning. In: Proceedings of the 24th Conference on Artificial Intelligence, AAAI, 2010, pp. 1306–1313. doi: 10.1145/1807406.1807449.
[8] G.A. Miller. WordNet: A lexical database for English. Communications of the ACM 38(11) (1995), 39–41. doi: 10.1145/219717.219748.
[9] S.P. Ponzetto, & M. Strube. Deriving a large-scale taxonomy from Wikipedia. In: Proceedings of the 22nd National Conference on Artificial Intelligence, AAAI, 2007, pp. 1440–1445. Available at: http://www.aaai.org/Papers/AAAI/2007/AAAI07-228.pdf.
[10] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, & Z. Ives. DBpedia: A nucleus for a web of open data. In: Proceedings of the 6th International Semantic Web Conference/Asian Semantic Web Conference, Springer, 2007, pp. 722–735. doi: 10.1007/978-3-540-76298-0_52.

[11] R. Speer, & C. Havasi. Representing general relational knowledge in ConceptNet 5. In: Proceedings of the 8th International Conference on Language Resources and Evaluation, 2012, pp. 3679–3686. Available at: http://www.lrec-conf.org/proceedings/lrec2012/pdf/1072_Paper.pdf.

[12] O. Etzioni, M. Cafarella, D. Downey, A.P. Shaked, S. Soderland, D.S. Weld, & A. Yates. Web-scale information extraction in knowitall: Preliminary results. In: Proceedings of the 13th International Conference on World Wide Web, ACM, 2004, pp. 100–110. doi: 10.1145/988672.988687.

[13] M.A. Hearst. Automatic acquisition of hyponyms from large text corpora. In: Proceedings of the 14th conference on Computational linguistics, ACL, 1992, pp. 539–545. doi: 10.3115/992133.992154.

[14] A. McCallum, & K. Nigam. A comparison of event models for naive bayes text classification. In: Proceedings of AAAI-98 Workshop on Learning for Text Categorization, AAAI, 1998, pp. 41–48.

[15] Z. Wang, H.X. Wang, J. Wen, & Y. Xiao. An inference approach to basic level of categorization. In: Proceedings of the 24th ACM International Conference on Information and Knowledge Management (CIKM), ACM, 2015, pp. 653–662. doi: 10.1145/2806416.2806533.

[16] B. Daille. Approche mixte pour l'extraction de terminologie: Statistique lexicale et filtres linguistiques. PhD dissertation, Université Paris VII, 1994.

[17] Z. Wang, K. Zhao, H. Wang, X. Meng, & J. Wen. Query understanding through knowledge-based conceptualization. In: Proceedings of the 24th International Joint Conference on Artificial Intelligence, AAAI, 2015, pp. 3264–3270.

[18] P. Li, H. Wang, K. Zhu, Z. Wang, & X. Wu. Computing term similarity by large probabilistic isA knowledge. In: Proceedings of the 22nd ACM International Conference on Conference on Information & Knowledge Management, ACM, 2013, pp. 1401–1410. doi: 10.1145/2505515.2505567.

[19] D. Marneffe, B. MacCartney, & C.D. Manning. Generating typed dependency parses from phrase structure parses. In: Proceedings of the 5th International Conference on Language Resources and Evaluations, 2006, pp. 449–454.

[20] D. Blei, A.Y. Ng, & M.I. Jordan. Latent dirichlet allocation. Journal of Machine Learning Research 3(Jan) (2003), 993–1022. Available at: http://jmlr.csail.mit.edu/papers/v3/blei03a.html.

[21] W. Hua, Z. Wang, H. Wang, K. Zheng, & X. Zhou. Short text understanding through lexical-semantic analysis. In: IEEE 31st International Conference on Data Engineering, IEEE, pp. 495–506. doi: 10.1109/ICDE.2015.7113309.

[22] J. Kupiec. Robust part-of-speech tagging using a hidden Markov model. Computer Speech & Language 6(3) (1992), 225–242. doi: 10.1016/0885-2308(92)90019-Z.

[23] Y. Wang, H. Li, H. Wang, & K.Q. Zhu. Toward topic search on the web. Technical report, Microsoft Research, 2010. Available at: https://www.microsoft.com/en-us/research/publication/toward-topic-search-on-the-web/?from=http%3A%2F%2Fresearch.microsoft.com%2Fapps%2Fpubs%2Fdefault.aspx%3Fid%3D166386.

[24] J. Wang, H. Wang, Z. Wang, & K.Q. Zhu. Understanding tables on the web. In: Proceedings of the 31st International Conference on Conceptual Modeling, Springer, 2012, pp. 1–14. doi: 10.1007/978-3-642-34002-4_11.

[25] F. Wang, Z. Wang, Z. Li, & J. Wen. Concept-based short text classification and ranking. In: Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, ACM, 2014. doi: 10.1145/2661829.2662067.

[26] P. Ferragina, & U. Scaiella. TAGME: On-the-fly annotation of short text fragments (by wikipedia entities). In: Proceedings of the 19th ACM International Conference on Information and Knowledge Management, ACM, 2010, pp. 1625–1628. doi: 10.1145/1871437.1871689.

[27] Y. Song, H. Wang, Z. Wang, H. Li, & W. Chen. Short text conceptualization using a probabilistic knowledgebase. In: Proceedings of the 22nd International Joint Conference on Artificial Intelligence, AAAI, 2011, pp. 2330–2336. doi: 10.5591/978-1-57735-516-8/IJCAI11-388.

[28] D. Kim, H. Wang, & A. Oh. Context-dependent conceptualization. In: Proceedings of the 23rd International Joint Conference on Artificial Intelligence, AAAI, 2013, pp. 2654–2661.

AUTHOR BIOGRAPHY

Lei Ji received her Bachelor and Master degrees from Beijing Institute of Technology and is currently a PhD student in computer science from Institute of Computing Technology, Chinese Academy of Sciences. She is currently a researcher at Microsoft Research Asia. Her research interests include knowledge graph, cross-modality visual and language learning.

Yujing Wang received her Bachelor and Master degrees from Peking University. She is currently a Researcher at Microsoft Research Asia. Her research interests include AutoML, text understanding and knowledge graph.

Botian Shi is currently a PhD student in computer science from Beijing Institute of Technology, who conducted this work during internship at MSRA. His research interests include natural language processing and multi-modal knowledge based image/video understanding.

Dawei Zhang received his Master degree from Peking University. He is currently the Founder of MIX Labs. His research interests include machine learning, natural language processing and knowledge graph.

Zhongyuan Wang received his Bachelor, Master, and PhD degrees in computer science at Renmin University of China in 2007, 2010, and 2015, respectively. He is currently a Senior Researcher and Senior Director of Meituan-Dianping. His research interests include knowledge base, Web data mining, online advertising, machine learning and natural language processing.

Jun Yan received his PhD degree in school of mathematical science at Peking University. He is currently the Chief AI Scientist of Yiducloud. His research interests include medical AI research, knowledge graph mining and online advertising.