Query Understanding through Knowledge-Based Conceptualization

Zhongyuan Wang, Kejun Zhao
Renmin University of China
Beijing, China
{kejunzhao, mengxf}@ruc.edu.cn

Haixun Wang
Microsoft Research
Beijing, China
zhy.wang@microsoft.com

Xiaofeng Meng, Ji-Rong Wen
Google Research
Mountain View, CA, USA
haixun@google.com
jirong.wen@gmail.com

Abstract

The goal of query conceptualization is to map instances in a query to concepts defined in a certain ontology or knowledge base. Queries usually do not observe the syntax of a written language, nor do they contain enough signals for statistical inference. However, the available context, i.e., the verbs related to the instances, the adjectives and attributes of the instances, do provide valuable clues to understand instances. In this paper, we first mine a variety of relations among terms from a large web corpus and map them to related concepts using a probabilistic knowledge base. Then, for a given query, we conceptualize terms in the query using a random walk based iterative algorithm. Finally, we examine our method on real data and compare it to representative previous methods. The experimental results show that our method achieves higher accuracy and efficiency in query conceptualization.

1 Introduction

We are concerned with the problem of conceptualizing short text such as search queries. Specifically, given a query, we are interested in infering the most likely concepts for terms in the query. Consider the example query `watch harry potter`. The term `harry potter` may refer to a variety of concepts including *book*, *movie*, and *character*. In this context, its most likely concept is *movie*. Short text conceptualization is important to a wide range of applications including text classification [Wang *et al.*, 2014a], head modifier detection [Wang *et al.*, 2014b], web table understanding [Wang *et al.*, 2012], query task identification [Hua *et al.*, 2013], etc.

There are many challenges within this problem. A document usually contains rich context which is crucial to lexical and syntactic disambiguation. For short text, however, neither parsing nor topic modeling works well because there are simply not enough signals in the input. To solve the problem, we must i) derive more signals from the input by combining it with external knowledge bases, and ii) devise a framework that enables the signals to fully interplay, so that we have more power to disambiguate and understand a short text.

- **Deriving signals from the input and external knowledge bases.** Humans usually do not have problems understanding short texts that are noisy, sparse, and ambiguous. It is, however, very difficult for machines. This is not surprising as most existing work treat text as bags of words [Blei *et al.*, 2003; Boyd-Graber *et al.*, 2007], and/or use statistical topic models [Phan *et al.*, 2008; Kim *et al.*, 2013] for sense disambiguation [Navigli, 2009; Moro *et al.*, 2014]. But the signals in the input might be too subtle for bag-of-words or co-occurrence based statistical approaches to capture. For example, the word `premiere` in `premiere Lincoln` is an important signal indicating `Lincoln` is a movie, and the word `watch` in `watch harry potter` indicates `harry potter` is a movie or a DVD (instead of a book). However, such lexical knowledge, i.e., `premiere` is an important attribute for a *movie* and `watch` usually takes a *movie* as a direct object, is not explicit in the input. We need extra knowledge bases to fill the gap.
- **Building a holistic model for short text understanding**. Existing natural language processing techniques adopt a multi-tiered model. For example, POS tagging is performed first, then chunking, parsing, entity resolution, etc. Signals flow from lower tiers to upper tiers, but not in the other direction. For short texts, we may need a new model. Take the query `watch harry potter` as an example. Here, `watch` is a verb because `harry potter` is a movie, and `harry potter` is a movie because `watch` is a verb. Thus, deciding the POS tag of a word actually requires signals from entity resolution, which happens at a much later tier in the NLP stack. Recent work on short text understanding [Hua *et al.*, 2015] has put more emphasis on using signals from lexical knowledge bases to assist query understanding, but it still uses a multi-tiered model that divides the task into three steps: text segmentation, word type detection, and instance disambiguation. A holistic model that allows all available signals to fully interplay in various subtasks will enable better understanding of short texts.

In this paper, we build a semantic network to enable us to derive more signals from the input. The knowledge we are interested in is knowledge of the language, or knowledge about how words interact with each other in a language (instead of encyclopedic knowledge). Such knowledge is important because the input often contains words that are not instances, but

Table 1: Example of Non-Instance Terms in Queries

Query	Non-Instance	Type	Instance
watch harry potter	watch	verb	harry potter
most dangerous python in the world	dangerous	adjective	python
population of china	population	attribute	china

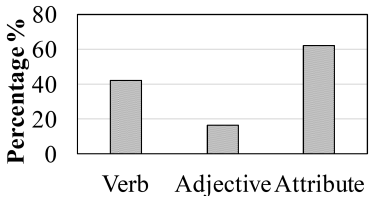


Figure 1: Queries Containing Non-Instance Terms

provide important signals to understand the instances. These include *verbs* that act on the instance, *adjectives* that modify the instance, or *attributes* of the instance. For example, the adjective `dangerous` in the query `most dangerous python in the world` strongly indicates that `python` is a *snake* instead of a *programming language*. More examples of such non-instance words are shown in Table 1.

Search queries contain many non-instance words. Figure 1 shows the percentage of queries that contain frequent non-instance words. In certain cases, these words actually represent instances (e.g., in `watch and jewelry`, `watch` is an instance), but in most cases they provide important signal to help disambiguate instances in the query.

In the rest of the paper, we address the following two challenges to understand short texts.

- We use a knowledge base that maps instances to their concepts, and build a knowledge base that maps non-instance words, including verbs and adjectives, to concepts. For example, `watch` maps to *movie*. But most words play more than one role: The word `watch` may be just a noun that maps to the concept *wrist watch*. One may think that an important signal for their role detection is POS tagging. Unfortunately, POS tagging does not perform well for queries due to their lack of linguistic structure. Table 2 shows some examples based on the Stanford Parser [Socher *et al.*, 2013]. Thus, we need additional signals to help make the decision. In this paper, we derive semantic signals from a large scale, data driven semantic network [Wu *et al.*, 2012], and also concept-level co-occurrence data, to solve this problem.
- Once we have access to a lot of semantic signals, we need a model that guides us toward a coherent understanding of the short text. We break the boundary between the tiers in natural language processing, and adopt a holistic framework. Specifically, we organize terms, concepts, and all relevant signals into a graph, and use an iterative random walk approach to reach a coherent understanding of the input.

The rest of the paper is organized as follows: Section 2 introduces some preliminary background, including that of a knowledge base we use in our work. Section 3 explains how we mine relationships between non-instance words and concepts. Section 4 presents our model for query conceptualization. Experimental studies are discussed in Section 5 and we conclude in Section 6.

Table 2: Bad NLP POS Tagging Results for Queries

Query	Result
adidas watch	adidas/NNS watch/VBP;
orange apple pie	orange/JJ apple/NN pie/NN
jump weekly anime	jump/VB weekly/JJ anime/NN
weekly anime jump	weekly/JJ anime/NN jump/NN

2 Preliminary

Humans can understand sparse, noisy, and ambiguous input such as short texts because they have knowledge of the language and the world. Many knowledge bases have emerged in recent years, including DBpedia[1], freebase [Bollacker *et al.*, 2008], Yago [Suchanek *et al.*, 2007], etc. Most of them are encyclopedic knowledge bases, containing facts such as Barack Obama's birthday and birthplace. They are essential for answering questions, but not for understanding them. To understand a question, we need knowledge of the language, for example, the knowledge that birthplace and birthday are properties of a person; and lexical knowledge bases are constructed for this purpose. In our work, we use a probabilistic lexical knowledge base known as Probase[2] [Wu *et al.*, 2012], but our techniques can be applied to other knowledge bases such as Yago.

Concepts

Probase contains millions of terms. Each term is a concept, an instance, or both. It also contains two major relationships between the terms: the isA relationship (e.g., `Barack Obama` isA *President*) and the isAttributeOf relationship (e.g., `population` isAttributeOf *country*). For an isA relationship between an instance e and a concept c, we can calculate the typicality of the concept given the instance as follows:

$$P^*(c|e) = \frac{n(e,c)}{\sum_{c_i} n(e,c_i)} \quad (1)$$

where $n(e,c)$ is the frequency we observe the isA relationship (e isA c) in a corpus. Typicality scores are critical for conceptualization or generalization.

Concept Clusters

As we mentioned, Probase contains millions of concepts, and a term may generalize into many concepts. For example, `tiger` maps to many concepts such as *animal*, *wild animal*, *exotic animal*, *jungle animal*, etc. Dimensionality reduction of the concept space is necessary to reduce computation complexity and to create more meaningful similarity functions, both of which are essential for inference.

We adopt a K-Medoids clustering algorithm [Li *et al.*, 2013] to group concepts into 5,000 disjoint concept clusters. For example, individual concepts such as *animal*, *wild animal*,

[1] http://wiki.dbpedia.org

[2] Probase data is available at http://probase.msra.cn/dataset.aspx

exotic animal, *jungle animal*, etc. are all grouped into the concept cluster *animal*. Thus, instead of conceptualizing an instance to individual concepts, we can conceptualize it to concept clusters. Specifically, the probability that an instance e maps to a concept cluster c is defined as

$$P(c|e) = \sum_{c^* \in c} P^*(c^*|e) \quad (2)$$

In the rest of the paper, we use concept to denote a concept cluster.

Attributes

We treat *attributes* as the first class citizen for short text processing. Probase contains the isAttributeOf relationship, which is derived from the following syntactic pattern:

the $\langle attr \rangle$ of (the/a/an) $\langle term \rangle$ (is/are/was/were/...)

Here, $\langle attr \rangle$ denotes the attribute to be extracted, $\langle term \rangle$ denotes either a concept (e.g., `country`) or an instance (e.g., `Italy`). As an example, from `the president of a country,` president is derived as an attribute of the concept *country*. Likewise, from `the capital of China`, capital is derived as an attribute of `China`. Then, because `China` belongs to the concept *country*, `capital` is also associated with *country*. Probase uses a RankSVM model to combine attributes derived from concepts and instances [Lee *et al.*, 2013]. In other words, it implements a function f to compute the following typicality score for attributes:

$$P(c|a) = f(n_{c^*,a}, n_{e_1,a}, \cdots, n_{e_k,a}) \quad (3)$$

where $n_{c^*,a}$ denotes how frequently a is derived as an attribute for a raw concept c^* and $n_{e_i,a}$ denotes how frequently a is derived as an attribute of e_i, which is an instance of concept c^*. We finally derive $P(c|a)$ by aggregating raw concepts into concept clusters.

Framework and Notation

Our framework consists of two parts: an offline part, which mines relationships between non-instance words and concepts, and an online part, which infers the concepts for terms in a query. The notions used in this paper are given in Table 3. A query contains one or multiple terms, and a term is a multi-word expression that can be a verb, an adjective, an attribute, or an instance. We use t to denote a term, and we use $P(z|t)$ to denote the type distribution of t, where type distribution means the probability that the term is a verb, an adjective, an attribute, or an instance.

3 Mining Lexical Relationships

In this section, we describe the offline process of mining lexical relationships. The knowledge we obtain is used in the online process of understanding short texts (Section 4).

Overview

Our goal is to obtain knowledge represented by the following two distributions:

Table 3: Basic Notions Used in this Paper

Notation	Meaning
c	a concept (cluster)
c^*	an individual concept in Probase
e	an instance
t	a term (t can be an instance, or a non-instance term)
term type	any of $\{verb, adjective, attribute, instance\}$
z	a random variable that denotes the term type
$P(z\|t)$	term t's type distribution
$P(t\|c, z)$	distribution of term t given its concept c and type z
$P(c\|t, z)$	distribution of concept c given term t and its type z

Table 4: Type distribution of a term $P(z|t)$

	Verb	Adjective	Attribute	instance
book	0.1033	0	0.0577	0.8389
watch	0.8374	0	0	0.1624
pink	0.0041	0.6830	0.0029	0.3101
harry potter	0	0	0	1

- $P(z|t)$: For a term t, $P(z|t)$ denotes the prior probability that t is of a particular type z. For instance, for the word `watch` that appears in web documents, we find that it is a $verb$ with probability $P(verb|\texttt{watch}) = 0.8374$.
- $P(c|t, z)$: For a term t of type z, $P(c|t, z)$ denotes the probability of the concept that the term is associated with. For example, $P(movie|\texttt{watch}, verb)$ denotes how likely the verb `watch` is related to the concept $movie$.

In the rest of this section, we describe how we obtain these distributions. We use these distributions for query understanding in Sec 4.

Parsing

To obtain the probabilities mentioned above, we first use an NLP parser to parse a large web corpus of billions of documents. Specifically, we use the Stanford Parser to obtain POS taggings and dependency relationships between tokens in the texts. The POS taggings reveal whether a token is an adjective or a verb, and the dependency between tokens, together with Probase as a lexicon, will be used to derive the dependency between adjectives/verbs and instances/concepts. We will describe more details in later part of this section.

Deriving $P(z|t)$

We compute $P(z|t)$ as follows:

$$P(z|t) = \frac{n(t, z)}{n(t)} \quad (4)$$

where $n(t, z)$ is the frequency term t appears as type z in the corpus, and $n(t)$ is the total frequency of term t. Table 4 shows some results.

Deriving $P(c|t, z)$

Since z can be *instance*, *attribute*, *verb*, or *adjective*, we discuss each case separately.

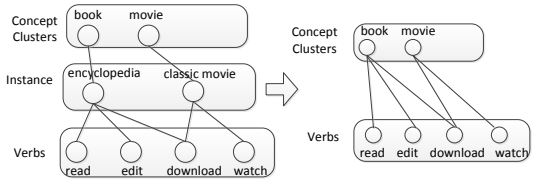


Figure 2: An Example of Bipartite Graph

Case 1: z **is** $instance$**.**
When z is $instance$, $P(c|t, z = instance)$ is reduced to $P(c|e)$, which can be derived from Probase using Eq. 2, as follows:

$$P(c|t, z = instance) = P(c|e) \tag{5}$$

Case 2: z **is** $attribute$**.**
When z is $attribute$, $P(c|t, z = instance)$ is reduced to $P(c|a)$, which can be derived from Probase using Eq. 3, as follows:

$$P(c|t, z = attribute) = P(c|a) \tag{6}$$

Case 3: z **is** $verb$ **or** $adjective$**.**
We first find the relationships between $verbs/adjectives$ and instances, and then use instances as a bridge, as shown in Fig. 2, to derive relationships between $verbs/adjectives$ and concepts.

Specifically, we detect co-occurrence relationships between instances, attributes, verbs, and adjectives in web documents parsed by the Stanford Parser. To obtain meaningful co-occurrence relationships, we require that the co-occurrence is embodied by dependency, rather than merely appearing in the same sentence. For example, for `The girl ate a big pear`, we obtain dependencies such as ($\texttt{eat}_{verb}$, $\texttt{pear}_{instance}$) and ($\texttt{big}_{adjective}$, $\texttt{pear}_{instance}$) from the Stanford parser. We then obtain $P(e|t, z)$, which denotes how likely a term t of type z co-occurs with instance e (through a dependency relationship):

$$P(e|t, z) = \frac{n_z(e, t)}{\sum_{e*} n_z(e*, t)} \tag{7}$$

where $n_z(e, t)$ is the frequency of term t and instance e form a dependency relationship when the type of t is z.

Then, using instances as the bridge, we obtain relationships between adjectives/verbs and concepts. More specifically, we have

$$\begin{aligned} P(c|t, verb) &= \sum_{e \in c} P(c, e|t, verb) \\ &= \sum_{e \in c} P(c|e, t, verb) \times P(e|t, verb) \\ &= \sum_{e \in c} P(c|e) \times P(e|t, verb) \end{aligned} \tag{8}$$

In the same spirit, we obtain $P(c|t, adjective)$ as follows:

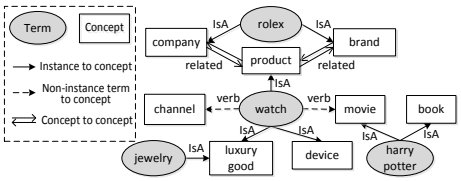


Figure 3: A semantic network centered around `watch`

$$P(c|t, adjective) = \sum_{e \in c} P(c|e) \times P(e|t, adjective) \tag{9}$$

where $P(c|e)$ is given by Eq. 2, $P(e|t, verb)$ and $P(e|t, attribute)$ are given by Eq. 7.

A Semantic Network

With the above efforts, we build a semantic network, which is a graph among terms that denote instances, concepts, attributes, adjectives, and verbs. Fig. 3 shows a subgraph of the semantic network centered around the term `watch` as an example.

As we can see, there are two types of nodes: One represents a *concept* (shown as a rectangle), and the other represents a *word* or a *term* (shown as an ellipse). Between these nodes, there are three types of edges: (1) the *isA* relationship between instances and concepts; (2) the *typicality* relationship between verbs/adjectives/attributes and concepts; and (3) the *relatedness* between two concepts.

We want to quantify the *strength* of each edge (relationship). There are two cases. In the first case, we quantify the strength of the relationship between a term to a concept, and we denote it as $P(c|t)$. This corresponds to the first two types of relationships mentioned above. It is important to be able to compute $P(c|t)$, and it is known as the generalization or conceptualization process. The second case, which corresponds to the third type of relation mentioned above, quantifies how strongly two concepts are related (for example, *product* and *company* are strongly related), and we denote it as $P(c_2|c_1)$.

- We assign a transition probability $P(c|t)$ to an edge from a non-concept term t (instance, attribute, verb, adjective) to a concept c:

$$P(c|t) = \sum_{z} P(c|t, z) \times P(z|t) \tag{10}$$

- We assign a transition probability $P(c_2|c_1)$ to an 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.

$$P(c_2|c_1) = \frac{\sum_{e_i \in c_1, e_j \in c_2} n(e_i, e_j)}{\sum_{c \in C} \sum_{e_i \in c_1, e_j \in c} n(e_i, e_j)} \tag{11}$$

The denominator normalizes the relatedness. In practice, we only take the top 25 related concepts for each concept ($P(c_2|c_1) = 0$ if c_2 is not among the top 25 related concepts of c_1).

4 Understanding Queries

In this section, we discuss how to annotate terms in a query with their proper concepts. For example, for query `apple ipad`, we want to annotate `apple` with *company* or *brand*, and `ipad` with *device* or *product*.

Our Approach

As described, the semantic network consists of terms, concepts, and their relationships (Figure 3). In particular, each term maps to a set of concepts. For a query q, the terms in q evoke a subgraph in the semantic network. For any term t in q, our goal is to find $\arg\max_c p(c|t, q)$. That is, we want to rank the concepts that t maps to in the given context of q.

Consider the query `watch harry potter` and Fig. 3, which contains the subgraph evoked by the query. Here, *movie* is a better concept than *book* for `harry potter` because *movie* can be reached by both `watch` and `harry potter`. A random walk based approach may be appropriate here to find the preferred concepts. However, traditional random walk methods are for simple networks where nodes, as well as edges, are homogeneous. In our case, the semantic network is not homogeneous. For example, `watch` can be a verb or an instance. In `watch harry potter`, the concept *movie* is related to both the verb sense of `watch` and the movie sense of `harry potter`. Once we are more certain that `watch` is a verb, we become more confident that `harry potter` refers to a *movie* instead of a *book*.

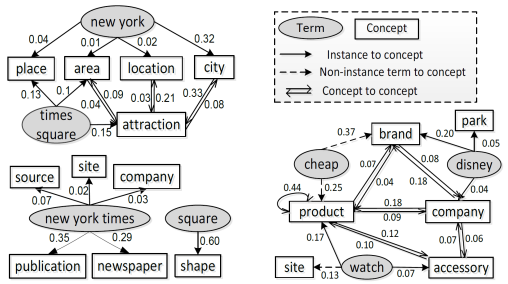
To address this problem, we use multiple rounds of random walks to find the most likely concepts. Within each round, we use our current belief of term-to-type and term-to-concept mappings to weight the candidate concepts. Between two rounds, we update our belief based on the new weights of the concepts. After the process converges, we obtain the final results.

Algorithm

The algorithm has 3 components. First, we segment the query into a set of candidate terms. Second, we create a graph out of the terms and their relationships. Finally, we use an iterative process to find the most likely mapping from terms to concepts.

Query Segmentation

We segment a query into a set of terms $T = \{t_1, t_2, \dots\}$. We use the Probase as our lexicon and identify all occurrences of terms in the query. For now, we only consider maximum terms, that is, terms not completely contained by other terms. For example, the query `angry bird` will be parsed into a single term `angry bird`, although `angry` and `bird` also belong to the lexicon. In some cases, the resulting terms may overlap. For example, for `new york times square`, we get two possible segmentations: 1) `new york` | `times square`, and 2) `new york times` | `square`. We consider both segmentations valid, and all of the terms will be used to construct the graph, as shown in Fig. 4(a). The most possible segmentation will be decided in the same process as we decide the concepts of the terms. In this case, since `new york` and `times square` reinforce concepts such as *area*, *location*, and *attraction*, while `new york times` and `square` do not reinforce any concrete concept, the former segmentation will be chosen at last. In query segmentation, we ignore terms such as prepositions (e.g., `of`, `for`, etc.) These terms provide useful linguistic clues to understand term dependency. However, this paper focuses on labeling terms by their concepts. Readers interested in the dependency structure of the query may refer to [Hua *et al.*, 2015].



(a) "new york times square" (b) "cheap disney watch"

Figure 4: Subgraphs of Example Queries

Graph Construction

The set of terms we obtain after segmentation evoke a subgraph of the semantic network we have constructed. More specifically, each term $t \in T$ connects to its concepts in the sematic network. The subgraph we are interested in is formed by the terms and their concepts. Figure 4 shows the subgraphs evoked by queries `new york times square` and `cheap disney watch`.

Random Walks

We perform multiple random walks to find the most likely concepts for each term. Each random walk consists of multiple iterations.

In the first random walk, let E denote the vector of the edge weights, and let V^n denote the vector of node weights in the n-th iteration of the random walk. In other words, the edge weights do not change between iterations, while the node weights change. Specifically, the weight of edge e in the first random walk is:

$$E[e] = \begin{cases} P(c|t) & e : t \to c \\ P(c_2|c_1) & e : c_1 \to c_2 \end{cases} \tag{12}$$

where $P(c|t)$ and $P(c_2|c_1)$ are derived by Eq. 10 and Eq. 11 respectively. The weight of node v at iteration 0 is ($|T|$ is the number of terms):

$$V^0[v] = \begin{cases} 1/|T| & v \text{ is a term} \\ 0 & v \text{ is a concept} \end{cases} \tag{13}$$

We use random walk with restart [Sun *et al.*, 2005] to update the weights of the nodes. Specifically, we have

$$V^n = (1 - \alpha)E' \times V^{n-1} + \alpha V^0 \tag{14}$$

where E' is the matrix form of E defined by Eq. 12. We perform the random walk for several iterations (as 2 iterations cover all relationships among terms and concepts in the evoked graph, there is no need to have many iterations).

Table 5: Non-instance Terms Conceptualization

Term		watch	book	pink	orange
Precision	GBIA	83.6%	89.7%	86.9%	78.1%
	NLP	60.9%	75%	61.7%	45.6%
Recall	GBIA	83.3%	83.3%	80.7%	80.6%
	NLP	73.1%	57.1%	85.4%	62.5%

After the current random walk, we obtain a new vector of node weights. For the example in Fig. 4(b), the weight of *product* becomes larger after the random walk. We then prepare the next random walk by creating a new vector of edge weights, that is, we update our belief about term-to-type and term-to-concept mapping. Specifically, we adjust E, the weights on edges, as follows:

$$E[e] \leftarrow (1-\beta) \times V^n[c] + \beta \times E[e] \quad e: t \rightarrow c \qquad (15)$$

Intuitively, in Fig. 4(b), because *product* can be reached by `cheap` and `disney` during the random walk, the weight of *product* becomes larger. Hence, we assign more weight to *watch-to-product* mapping and less weight to *watch-to-site* mapping.

With the new node weight vector and the new edge weight vector, we start the next random walk. The process is repeated till convergence. We argue that our algorithm converges as it is known that a standard random walk with restart converges [Fujiwara *et al.*, 2012]. Specifically, the convergence of Eq 14 is guaranteed when E is constant [Strang, 2003]. In our case, since E and V are non-negative and $E \propto V^n$, it follows that the entire process will converge.

At last, we annotate concepts of a term in the given query by normalizing its edges' weights:

$$p(c|t,q) = \frac{E(t \rightarrow c)}{\sum_{c_i} E(t \rightarrow c_i)} \quad t \in q \qquad (16)$$

5 Experiment

We create two labeled datasets out of randomly selected search queries. The first consists of 600 queries about 6 ambiguous terms: `watch`, `book`, `pink`, `orange`, `population`, `birthday` (100 queries for each term). The second also consists of 600 queries, among which 200 contain ambiguous terms `apple` or `fox` and the other 400 are totally random. We ask 12 colleagues to label the 1200 queries. It is easy to label the type of a term in each query. For concept labels, we run our algorithms first, and ask our volunteers to rank the top-N concepts for each term. Specifically, a concept is assigned a score $rel_i = 1/\frac{2}{3}/\frac{1}{3}/0$ if it is considered a correct/related/not-sure/false label. We use the precision of Top-N concepts [Lee *et al.*, 2013] for evaluation, i.e., $Precision@N = \frac{\sum_i^N rel_i}{N}$.

We evaluate our algorithm (denoted as **GBIA** or **G**raph-**B**ased **I**terative **A**lgorithm) in several aspects. We first study precision and recall of type detection (i.e., detecting whether a term is an instance, attribute, verb, or adjective) on the 1200 queries. We compare them with labels given by the Stanford parser (denoted as **NLP**). Table 5 shows the outcome, and our method has a clear advantage. As mentioned previously, our type detection method is based on lexical relationships

Table 6: Lexical Relationships of Non-instance Terms

read (verb)	dangerous (adj.)	population (attribute)
passage	crime	non-financial factor
anime	emergency	country
book	disaster	city
novel	law	outlying area
writer/author	disease	school
magazine	snake	mammal
writing	magazine	insect
blogs	therapy	foreigner
fairytales/story	drug	threat
memoir	sound	organisms

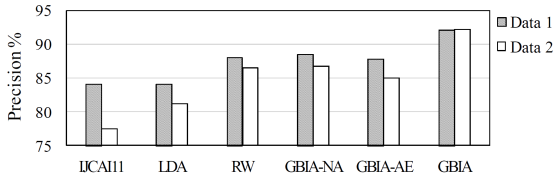


Figure 5: Precision of Query Conceptualization

mined from the web corpus. We also manually evaluate the quality of the mined relationships by inspecting the lexical relationships of several non-instance terms. We use Stanford Parser to parse 400 million sentences, which gives us POS taggings and dependency relationships. Then, we use Eq 6, 8, and 9 to rank concepts for each non-instance term. Table 6 shows the top 10 concepts generated for the chosen non-instance terms. They agree with manual results.

Next, we evaluate term conceptualization. We compare our method with the following 5 methods (2 state-of-the-art approaches and 3 variants of our method).

- **IJCAI11** [Song *et al.*, 2011]. It groups instances by their conceptual similarity, and uses simple Bayesian analysis to conceptualize each group.
- **LDA** [Kim *et al.*, 2013]. It combines LDA (which mostly models co-occurrence relationships) and Probase (which mostly models isA relationships) for short text conceptualization.
- **RW** (random walk). This is a variant of our method. It is a pure random walk approach without adjusting the weights on edges during the whole process.
- **GBIA-NA**. This is a variant of our method. It omits all non-instance terms in the queries, i.e. if a term appears in our collected non-instance term list, we remove it from the conceptual graph.
- **GBIA-AE**. This is a variant of our method. It treats all terms in Probase instance list as instances, i.e., if a verb/adjective is also an instance in Probase, we initialize its type to be an instance only.

Figure 5 shows the overall precision of the top-2 concepts. The overall precision is calculated as $\frac{3}{4}Precision@1 + \frac{1}{4}Precision@2$. We can see that our method achieves the highest precision on both datasets. Table 7 shows some examples of the results.

We also examine the number of iterations and the time requirement of our method. Figure 6 shows that the efficiency

Table 7: Query Conceptualization Example

Query	Non-instance	Instance
watch harry potter youtube	watch:verb	harry potter:movie,fandoms youtube:website,network
buy watch and jewellery	buy:verb	watch:product,accessory jewellery:product,accessory
how to bake an apple	bake:verb	apple:fruit,food
tim cook apple ceo	ceo:attribute	tim cook:executive,leader apple:company,brand ceo:senior executive,leader
yummy orange	yummy:adj	orange:fruit,food
orange t shirt dress	orange:adj	t shirt:garment,product dress:garment,product

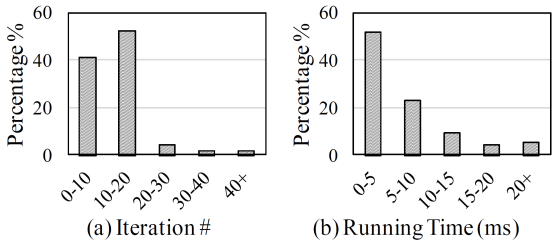


Figure 6: Algorithm Efficiency

of our approach is acceptable for online search, with the Y-axis representing the percentage of queries in the 1200 queries.

6 Conclusion

Query understanding is a challenging task. We have built a lexical knowledge base to discover fine-grained semantic signals from the input, and introduce a new graph-based iterative framework to determine the type as well as the concepts of the terms in the query. Experiments on real data have shown that our method achieved great improvement over previous methods for query understanding.

7 Acknowledgments

This work was partially supported by the National Key Basic Research Program (973 Program) of China under grant No. 2014CB340403, the National 863 High-tech Program (No. 2013AA013204), the Natural Science Foundation of China (No. 61379050, 91224008), and the Fundamental Research Funds for the Central Universities & the Research Funds of Renmin University of China.

References

[Blei *et al.*, 2003] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[Bollacker *et al.*, 2008] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. ACM, 2008.

[Boyd-Graber *et al.*, 2007] Jordan L. Boyd-Graber, David M. Blei, and Xiaojin Zhu. A topic model for word sense disambiguation. In *EMNLP-CoNLL*, pages 1024–1033, 2007.

[Fujiwara *et al.*, 2012] Yasuhiro Fujiwara, Makoto Nakatsuji, Makoto Onizuka, and Masaru Kitsuregawa. Fast and exact top-k search for random walk with restart. *Proceedings of the VLDB Endowment*, 5(5):442–453, 2012.

[Hua *et al.*, 2013] Wen Hua, Yangqiu Song, Haixun Wang, and Xiaofang Zhou. Identifying users' topical tasks in web search. In *WSDM*, pages 93–102, 2013.

[Hua *et al.*, 2015] Wen Hua, Zhongyuan Wang, Haixun Wang, Kai Zheng, and Xiaofang Zhou. Short text understanding through lexical-semantic analysis. In *International Conference on Data Engineering (ICDE)*, April 2015.

[Kim *et al.*, 2013] Dongwoo Kim, Haixun Wang, and Alice Oh. Context-dependent conceptualization. In *IJCAI*, 2013.

[Lee *et al.*, 2013] Taesung Lee, Zhongyuan Wang, Haixun Wang, and Seung won Hwang. Attribute extraction and scoring:a probabilistic approach. In *ICDE Conference*, 2013.

[Li *et al.*, 2013] Peipei Li, Haixun Wang, Kenny Q. Zhu, Zhongyuan Wang, and Xindong Wu. Computing term similarity by large probabilistic isa knowledge. In *ACM International Conference on Information and Knowledge Management (CIKM)*, 2013.

[Moro *et al.*, 2014] Andrea Moro, Alessandro Raganato, and Roberto Navigli. Entity Linking meets Word Sense Disambiguation: a Unified Approach. *Transactions of the Association for Computational Linguistics (TACL)*, 2:231–244, 2014.

[Navigli, 2009] Roberto Navigli. Word sense disambiguation: A survey. *ACM Computing Surveys (CSUR)*, 41(2):10, 2009.

[Phan *et al.*, 2008] Xuan Hieu Phan, Minh Le Nguyen, and Susumu Horiguchi. Learning to classify short and sparse text & web with hidden topics from large-scale data collections. In *WWW*, pages 91–100, 2008.

[Socher *et al.*, 2013] Richard Socher, John Bauer, Christopher D Manning, and Andrew Y Ng. Parsing with compositional vector grammars. In *In Proceedings of the ACL conference*. Citeseer, 2013.

[Song *et al.*, 2011] Yangqiu Song, Haixun Wang, Zhongyuan Wang, Hongsong Li, and Weizhu Chen. Short text conceptualization using a probabilistic knowledgebase. In *IJCAI*, pages 2330–2336, 2011.

[Strang, 2003] Gilbert Strang. Introduction to linear algebra. *Cambridge Publication*, 2003.

[Suchanek *et al.*, 2007] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706. ACM, 2007.

[Sun *et al.*, 2005] Jimeng Sun, Huiming Qu, Deepayan Chakrabarti, and Christos Faloutsos. Neighborhood formation and anomaly detection in bipartite graphs. In *ICDM*. IEEE, 2005.

[Wang *et al.*, 2012] Jingjing Wang, Haixun Wang, Zhongyuan Wang, and Kenny Zhu. Understanding tables on the web. In *International Conference on Conceptual Modeling*, October 2012.

[Wang *et al.*, 2014a] Fang Wang, Zhongyuan Wang, Zhoujun Li, and Ji-Rong Wen. Concept-based short text classification and ranking. In *ACM International Conference on Information and Knowledge Management (CIKM)*, October 2014.

[Wang *et al.*, 2014b] Zhongyuan Wang, Haixun Wang, and Zhirui Hu. Head, modifier, and constraint detection in short texts. In *International Conference on Data Engineering (ICDE)*, 2014.

[Wu *et al.*, 2012] Wentao Wu, Hongsong Li, Haixun Wang, and Kenny Qili Zhu. Probase: a probabilistic taxonomy for text understanding. In *SIGMOD Conference*, pages 481–492, 2012.