How to Make a Semantic Network Probabilistic

Zhongyuan Wang [†,1] Haixun Wang [‡,2] Yanghua Xiao [#,3] Ji-Rong Wen [♮,4]

[†]*Microsoft Research, Beijing, China* [‡]*Google Research, Mountain View, CA, USA* [#]*Fudan University, Shanghai, China* [♮]*Renmin University of China, Beijing, China*

[1]zhy.wang@microsoft.com [2]haixun@google.com [3]shawyh@fudan.edu.cn [4]jirong.wen@gmail.com

ABSTRACT

Words and phrases associate with each other to form a semantic network. Characterizing such associations is a first step toward understanding natural languages for machines. Psychologists and linguists have used concepts such as typicality and basic level conceptualization to characterize such associations. However, how to quantify such concepts is an open problem. Recently, much work has focused on constructing semantic networks from web scale text corpora, which makes it possible for the first time to analyze such networks using a data driven approach. In this paper, we introduce measures such as *typicality*, *basic level conceptualization*, *vagueness*, *ambiguity*, and *similarity* to systematically characterize the associations in a semantic network. We use such measures as the basis for probabilistic semantic inferencing, which enables a wide range of applications such as word sense disambiguation and short text understanding. We conduct extensive experiments to show the effectiveness of the models and the measures we introduce for the semantic network.

1. INTRODUCTION

Natural language understanding is always the eternal quest for machines. Human beings can easily understand natural language because they have rich background knowledge in their mind. To fill this gap, some researchers propose implicit knowledge mining, such as PLSI [23], LDA [10], word embedding for DNN [18, 31], etc. On the other hand, with big data and community efforts, explicit knowledge mining becomes possible and practical for machines. Recent years, lots of efforts are devoted to constructing knowledgebases, such as Freebase [12] and Yago [41]. Most of these knowledgebases contain millions of entities (with unique ids) and billions of facts among these entities. Facts in these knowledgebases are usually black or white (E.g. <Barack Obama [/m/02mjmr]; date_of_birth; August 4, 1961>). With these knowledge facts, machines can answer "fact lookup questions" in search engines. However, a key challenge here is that machines should understand questions first. Currently, to bypass this challenge and leverage knowledge facts, major search engines prepare a white list of mapping between questions and knowledge facts by rule-based or mining-based approaches.

But human beings can understand questions easily. For a 10 years old child, even if he/she doesn't have billions of facts in mind, he/she can understand the questions correctly. This is because human beings have common sense in mind, and can do some reasoning when they read the text. Therefore, when users launch queries such as "band for wedding" (a kind of service), they may laugh at search engines when they see search results and ads containing "wedding band" (a kind of ring) on the return page.

Recently, major search engines realized the shortage of keyword-based search systems and began to integrate semantics to their search and ads systems. One of the effective practices is text annotation or conceptualization. That is, given a short text, the goal is to infer concepts in the text. So that machines can understand that "apple ipad" is a kind of device, while "apple pie" is a kind of dessert. They are quite different though they are similar literally. This conceptualization-based similarity comparison has become an important signal in search engines (details in subsection 5.4).

However, there are many challenges in short text conceptualization. Consider the following concrete search queries:

1. *"New York"*
2. *"china, india, brazil"*
3. *"weather april in paris"*
4. *"april in paris lyrics"*

Making machines understand these queries requires prior probability, mutual reasoning, and context leverage:

- Without prior probability, machines map "New York" to *state*, *city*, *movie*, and *song* equally. But this is not human thinking. People consider that "New York" as *state* or *city* are much more typical and representative than it as *movie* or *song*.
- Given "china", it belongs to *country* and *fragile item*. With its context "india", human beings can infer that they belong to *country*. But by adding "brazil", *emerging market* or *BRICS* is a better concept to model these terms.
- Typically, "april in paris" is a *song* in human mind when it appears alone or with context "listen to" or "lyrics." But in the example of "weather april in paris", since *song* and *weather* have not obvious relations, "april in paris" should be divided into two instances "april" and "paris."
- Mapping "weather" to its hypernym *factor*, *variable*, *information*, *topic*, *condition*, *environmental factor*, and so on. But some of these concepts are meaningless actually, and they may introduce noise to results.

Hence one can see that understanding short texts is challenging for machines. But human beings can understand these texts easily, though they do not have billions of knowledge facts in mind. This means: 1) common sense knowledge is more important for text understanding; 2) knowledge in human mind is probabilistic.

In this paper, we call those systems which focus on extracting facts among *entities* as **knowledgebases** (such as Freebase [12] and Yago [41]), and those systems which focus on common sense relations among *terms* as **semantic networks** (such as KnowItAll [20] and Probase [42]).

To overcome above challenging problems in text understanding, we need a probabilistic semantic network. This probabilistic semantic network should associate different scores with nodes (such as concepts and instances) and their edges (such as isA relations, similarity relations). In this paper, we will focus on deriving these essential scores and probabilities in a semantic network: *Typicality, BLC (Basic level conceptualization), Vagueness, Ambiguity,* and *Similarity*. We show their definitions and examples in Table 1:

- *Typicality* is the basic score in any semantic network. It describes the typical grade of an instance e given a concept c, or reverse. With this score, we can know "dog" and "cat" are more typical than "starfish" in the concept *animal*. Table 1 gives more examples ranked by typicality score for a given concept *animal* or an instance *penguin*. The insight of this score is discussed in subsection 4.1.

- *BLC (Basic level conceptualization)* provides the ability of finding an appropriate level of concepts in a set of hierarchically organized concepts for a given instance e. Unlike *typicality*, concepts ranked by *BLC* score are a trade-off between general concepts and specific concepts. It can be also treated as a compromise between the accuracy of classification and the power of prediction (please refer to subsection 4.2.1). Besides, we compare *BLC* with PMI and $commute\ time$ theoretically, and show the similarity and difference among them. We give more details in subsection 4.2.

- We also define *Vagueness* and *Ambiguity* for terms in the semantic network. The former corresponds to concepts, and the latter corresponds to instances. With these two scores, we can resolve part of issues mentioned in above examples. For example, we know "item" is a vague concept, and ignore it in the text understanding. We also recognize the "apple" is an ambiguous instance, and do sense disambiguation with its context. These two measures will be described in subsection 4.3 and 4.4 respectively.

- Measuring semantic *Similarity* between terms (i.e., words or multi-word expressions) is another fundamental problem in the semantic network. With this score, we can know "global company" and "multinational corporation" are similar, while "apple ipad" and "apple pie" are not. This is critical to many text understanding tasks. The detailed technique of *similarity* calculation will be discussed in subsection 4.5.

With above scores, machines can be more "smart" in the short text understanding. Let's recall above concrete examples. With typicality scores, machines know that "New York" is not a typical *movie*; with BLC scores, machines know that *state* and *city* are more representative than *movie* and *song* for the term "New York;" with vagueness scores, machines can remove *factor*, *variable*, *information*, and *topic* from concepts of "weather," then machines know that *song* has not obvious relations with *environmental factor*, but *city* has, so that "april in paris" should be divided into "april" and "paris;" with ambiguity measures, machines know "apple" is an ambiguous term; with similarity scores, machine find that "apple" and "microsoft" are very similar, and "microsoft" can be used to distinguish the sense of "apple."

Therefore, all of above scores are essential components in the semantic network. They are the foundation when we try to leverage the semantic network in real applications. In this paper, we will show a methodology of deriving these scores and weights in a general semantic network, and how they enable semantic inferencing.

The rest of the paper is organized as follows. Section 2 introduces the concept of semantic network. Section 3 shows how we cluster concepts and mine hierarchies in the semantic network. Section 4 presents details of scores we proposed. Section 5 gives experiment results and compares our approaches with other methods. We discuss related work in Section 6 and conclude in Section 7.

2. SEMANTIC NETWORKS

Lots of knowledgebases put their efforts on acquiring as many facts as possible. E.g. Freebase claims it has 1.2 billion fact triples (the version of freebase-rdf-2013-08-26-test.gz), and Yago claims it has 120 million facts about entities. Both of these knowledgebases ignore the following key problems:

1. Knowledge in human minds is composed of terms, instead of entity ids.
2. Knowledge in human minds usually are not black or white. It associates with probability.

Because of the above problems, this kind of knowledgebases is limited in lots of applications. As mentioned in the Introduction section, machines should first build mapping such as entity linking between queries and knowledge facts, then they can use these knowledgebases to answer a small portion of queries.

On the other hand, term-based semantic networks such as KnowItAll, NELL, and Probase try to capture general knowledge in human minds. They hope to enable machines to better understand human communication, not just a small portion of short texts. These semantic networks are natural language oriented, and usually associate with some statistical information such as the number of co-occurrence. In this paper, we will show how to use this statistical information, to make the semantic network probabilistic, and derive scores we describe in the Introduction section. We will take Probase[1] [42] as a running example in this paper. Definitely, our techniques can be applied to other semantic networks.

Probase is acquired from 1.68 billion web pages. It consists of the *isa* relations extracted from sentences matching the Hearst patterns [22]. For example, from the sentence "... presidents such as Obama ...", it extracts a piece of evidence for the claim that "Obama" is an instance of the concept *president*. The core version of Probase contains 3,024,814 unique concepts, 6,768,623 unique instances, and 29,625,920 edges among them.

This kind of term-based semantic networks is naturally similar as knowledge in human mind. However, they are also noisy and ambiguous. In the following sections, we will show how to scoring the terms and relations in the semantic network, to make them usable for knowledge empowered applications.

3. CONCEPT CLUSTERS AND HIERARCHIES

Some semantic networks such as Probase contain millions of fine-grained concepts. These are the concepts used by humans, and

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

Score Type	Definition	Examples
Typicality	The typical grade of an instance e given a concept c, or vice versa.	animal$_c$ $\rightarrow$ {*dog, cat, horse, bird, rabbit, deer, cow, sheep, goat* ... } penguin$_e$ $\rightarrow$ {*animal, bird, species, flightless bird, character, brand* ... }
BLC	Given an instance e, an appropriate level of concepts in a set of hierarchically organized concepts.	penguin$_e$ $\rightarrow$ {*flightless bird, cold-dependent animal species, colorful criminal, deep-diving bird, winter animal, polar animal* ... }
Vagueness	Given a concept c, a measure of its uncertain, indefinite, or unclear character or meaning.	Vague concepts: { entity, phrase, type, item, thing, result, keyword, clue, way, term, ... }
Ambiguity	Given an instance e, its ambiguous level.	Level 0 (unambiguous): {alcohol, computer, coffee, potato, bean, ... } Level 1 (borderline): {nike, google, facebook, twitter, xbox, kindle, ... } Level 2 (ambiguous): {jordan, fox, puma, python, apple, jaguar, ... }
Similarity	The similarity between two terms.	<tiger, jaguar>: 0.979; <caged animal; game animal>: 0.996; <animal, poodle>: 0.720; <banana, beef>: 0.007; <apple, ipad>: 0.006; ...

Table 1: Different Score Types and Examples

it is our hope that they can empower machines to understand texts in natural language. Indeed, Gregory Murphy [33] claimed *"Concepts are the glue that holds our mental world together,"* and Nature magazine book review also says *"Without concepts, there would be no mental world in the first place."* [11]

However, these millions of concepts are usually not independent of each other. Instead, they interact with each other and form intricate structures.

3.1 Overview

In this paper, we focus on two relationships among concepts:

- Concepts that overlap. Concepts are not orthogonal, e.g. *country* and *developing country* have many overlapping instances. Some concepts, e.g. *country* and *nation*, are synonyms, meaning their instances are almost exactly the same.
- Concepts that exhibit isA relationships. For instance, *fruit* isA *food*. The isA relationship is the "backbone" relationship in the semantic network, and it enables us to generalize or conceptualize.

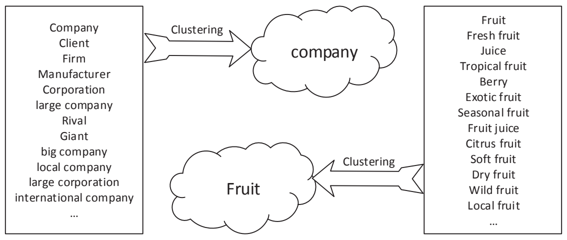


Figure 1: Concept Clusters

For the first relationship, we use a clustering algorithm to group concepts based on their instance distributions. Figure 1 gives two examples of concept clustering results. As we can see, *company*, *client*, *firm*, *manufacturer*, and *corporation*, these concepts have similar meanings, and they will be put into the same cluster. Besides, *company*, *large company*, and *international company* are also put into the same cluster because they have many overlapping instances. The cluster *Fruit* shows similar features.

We can benefit a lot from concept clustering: First, we reduce the concept dimensions from millions to a few thousands, which improves runtime performance; Second, based on the concept clusters, it is easier to build a concept hierarchy for reasoning and sense mining; Third, concept clusters are important for sense disambiguation.

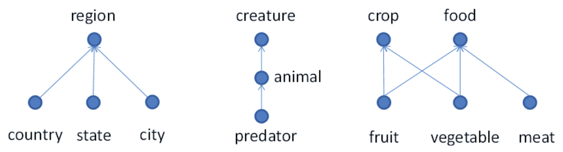


Figure 2: Concept Cluster Hierarchies

For the second relationship (isA relationships), it is twofold. One is the isA relation between concepts, the other is the isA relation between concept clusters. The former relationship is natural since the semantic network is constructed from it. It can be directly used in all kinds of scoring. The latter relationship can be built upon original isA relations among concepts (details will be discussed in subsection 3.3). Figure 2 shows several concept cluster hierarchies. E.g. the cluster *predator* belongs to *animal*, and *animal* is a kind of *creature*.

Another option to mine these two relationships is using a hierarchical clustering algorithm. The reasons that we adopt this 2-phase approach instead of hierarchical clustering methods are as follows: first, the hierarchical clustering method may suffer from its poor merge or split decisions, which make the results out of control; second, the hierarchical clustering method is not easy to be applied for millions of nodes.

3.2 Mining Concept Clusters

In this subsection, we introduce how to cluster millions of concepts. Considering the effectiveness and efficiency of clustering on millions of nodes, we employ a k-Medoids clustering algorithm [27] to cluster these concepts.

The basic idea is that if two concepts share many entities, they are similar to each other. Therefore, we use the instance distribution to represent each concept and evaluate the semantic distance between two concepts c_1 and c_2 by Eq. (1)[2].

$$D_{clustering}(c_1, c_2) = 1 - cosine(\mathcal{I}_{c_1}, \mathcal{I}_{c_2}) \quad (1)$$

where $\mathcal{I}_{c_i}$ represents the vector of instance distribution of concept c_i as defined in Eq. (2).

$$\mathcal{I}_c = \langle (e_1, f_1), \cdots, (e_{|c|}, f_{|c|}) \rangle \quad (2)$$

[2]Our experiments reveal that Cosine outperforms other similarity/ distance evaluation functions, such as Jaccard, JaccardExtended, Jensen-Shannon, and the KL divergence in the semantic network.

In each component (e, f) of $\mathcal{I}_c$, e is the instance, and f is the co-occurrence of e and c in Hearst patterns' [22] sentences.

Then we need to choose initial centers and set the number of clusters k. Good initial centers are essential for the success of k-Medoids clustering algorithm. Instead of using random initial centers, we identify good initial centers incrementally [32]. The first center is randomly selected among all candidate concepts. Then we want to select a concept farthest from existing centers as the next initial center. That is, for each remaining concept, we set its minimum distance with existing centers as its "weight," and select the concept with maximum "weight" as the next center, i.e.,

$$m = \{m_i | \max_{c_j}\{\min_i\{D_{clustering}(m_i, c_j)\}\} > \alpha\} \tag{3}$$

where m_i is the existing center, c_j is the candidate concept, and α is a distance threshold that the new center has to meet. Clearly, the number of clusters k is determined by the threshold α. Based on our experiment results, we set α = 0.7 as an optimal value.

According to the above processing of k-Medoids, we can get k clusters for all given concepts.

3.3 Mining Concept Cluster Hierarchies

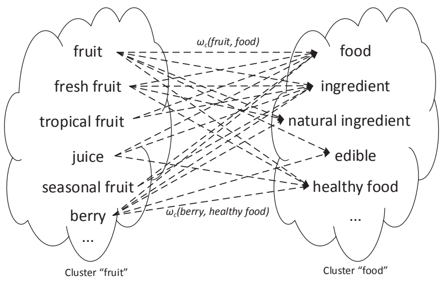


Figure 3: Concept Clusters Hierarchy

After concept clustering, we reduce millions of concepts to thousands of concept clusters. Then we can mine isA relations among these clusters according to original isA relations in the semantic network. As Fig. 3 shows, since *fresh fruit* and *juice* belong to *ingredient* and *healthy food*, we add the isA relation between the cluster "fruit" and cluster "food," and the weight of this edge is the sum of original weights in the semantic network as follows:

$$\omega_{Cl}(Cl_\alpha, Cl_\beta) = \sum_{c_\mu \in Cl_\alpha, c_\nu \in Cl_\beta} \omega_c(c_\mu, c_\nu) \tag{4}$$

where c_μ and c_ν are concepts, Cl_α and Cl_β are concept clusters, ω_c is the raw weight between concepts, and ω_{Cl} is the derived weight between concept clusters.

Definitely, original isA relations between clusters may conflict. In this case, these relations weaken themselves mutually, and cause ω_{Cl} small. We just let it go since this reflect the actual case, and further scoring can be applied based on these new weights.

With above efforts, we build an abstract semantic network with thousands of concept clusters, and millions of instances. This abstract semantic network is complementary to original semantic network. All the scores and weights we introduce in the following section can be applied to this abstract semantic network.

This abstract semantic network can benefit the efficiency of semantic network based applications. It is also useful for ambiguity and similarity calculation. The details will be discussed in subsection 4.4 and 4.5 respectively.

4. SCORING THE SEMANTIC NETWORK

In this section, we describe how we derive the five fundamental scores we introduced in the Introduction section in a large data driven semantic network.

4.1 Typicality

One of the most basic scores is the **typicality** score. According to a study in cognitive science and psychology [33], each category (concept) might have a most typical item, which is perhaps an average or ideal example that people extract from seeing real examples. This widely exists in human minds. E.g. given a concept "*bird*," more people may think of "robin" instead of "penguin." Similarly, given a concept "*country*," more people may think of "USA" or "China" instead of "Seychelles." It looks like human beings assign a typicality score for each instance in a concept, and ranked them automatically when they think of the concept.

DEFINITION 4.1 (TYPICALITY). *Typicality is a graded phenomenon, in which items can be extremely typical (close to the prototype), moderately typical (fairly close), atypical (not close), and finally borderline category members (things that are about equally distant from two different prototypes) [33]. Specifically, given a concept c in the semantic network, typicality $P(e|c)$ is the typical grade of an instance e in this concept. Similarly, given an instance e in the semantic network, typicality $P(c|e)$ is the typical grade of a concept c for this instance.*

Typicality score is very useful for machines. It can make machines do reasoning like human beings. Without typicality scores, knowledge facts in the semantic network are not easy to use. E.g. for the term "apple," it may belongs to lots of categories, such as *fruit, company, book, movie,* and *music track*. For human beings, probably they will think of *fruit* or *company* when they see "apple." For machines, if they do not have typicality scores, they will treat *music track* as important as *fruit* or *company*. This makes machines cannot reason over the semantic network as human beings. What is worse, according to our observations, all kinds of terms or phrases may be names of *book*, *movie*, or *music track*, etc. Without typicality scores, machines will think all texts are related, because they all belong to these concepts. This leads to many errors when machines understand texts.

However, how to compute the typicality is an interesting and challenging problem. Mervis et al. [30] found that simple frequency of an item's name did not predict its typicality. For example, "chicken" is very frequently talked-about (E.g. its frequency is 7,084 in the semantic network), and it's a kind of bird. But it's not considered as typical as some less frequently encountered and discussed birds, such as "robin" (E.g. its frequency is 537). But how often an item is thought of as being a member of the category can measure the typicality [9]. For example, by levering Hearst patterns [22], we find "chicken" is thought of as being a member of *bird* with 130 times, while "robin" as a *bird* with 279 times in a corpus of web documents. It is consistent with the fact the "robin" is more typical than "chicken" as a bird.

Intuitively, typicality score can be driven from co-occurrences of concept and instance pairs as follows:

$$\begin{aligned} P(e|c) &= \frac{n(c,e)}{\sum_{e_i \in c} n(c,e_i)} \\ P(c|e) &= \frac{n(c,e)}{\sum_{e \in c_i} n(c_i,e)} \end{aligned} \tag{5}$$

where $n(c, e)$ is the co-occurrence of concept c and instance e in Hearst patterns' sentences from the whole Web documents.

4.2 Basic Level Conceptualization

With the typicality score, machines can do some simple reasoning, such as conceptualization [40, 24]. Basically, given an instance e, conceptualization tries to map it to some concepts. But sometimes, the typical concepts are not the representative concepts of instances. In this section, we will discuss how to do the basic level conceptualization for a given instance.

4.2.1 Definition

For any instance, its concepts can be thought of as a set of hierarchically organized categories, ranging from extremely general to extremely specific [33]. Consider the term "jewelry." It can be categorized into a large number of concepts, including *item*, *valuable*, *valuable item*, *handmade and commercial craft item*, etc. Consider these two concepts:

1. item
2. handmade and commercial craft item

Here, $c =$*item* has high typicality $p(c|e)$, that is, given $e =$ *jewelry*, it is highly likely that the concept of *item* comes into mind. On the other hand, $c =$*handmade and commercial craft item* has high typicality $p(e|c)$, that is, when people talk about *handmade and commercial craft item*, very likely they mean *jewelry*. Thus, they represent two extremes: *item* is a very general concept for *jewelry*, while *handmade and commercial craft item* is a very specific one. These two extremes have following two characteristics:

- From the perspective of classification, general concepts tend to maximizes accuracy of classification. E.g. if we classify instances to *item*, it may be most likely correct.
- From the perspective of prediction, specific concepts tend to allow for greater power in prediction. E.g. *handmade and commercial craft item* is able to predict more about *jewelry* (its properties, its appearance) than *item*.

In other words, general concepts may be correct answers to a given instance, but they cannot distinguish different kinds of instances and can also distort the meaning of instance. On the other hand, specific concepts preserve more useful information about instances, but their coverage is limited. In some scenarios, we want to find the most appropriate concepts which are not too general nor too specific. We use **BLC (Basic level conceptualization)** to capture this feature.

DEFINITION 4.2 (BASIC LEVEL CONCEPTUALIZATION). *Given an instance e in the semantic network, of all possible concepts in a hierarchy to which e belongs, the appropriate level of concepts is the most natural, preferred level at which to conceptually carve up the world.*

In other words, BLC concepts are a trade-off between general concepts and specific concepts. It can be also treated as a compromise between the accuracy of classification and the power of prediction.

4.2.2 BLC Using Typicality with Smoothing

The typicality score defined in Equation 5 can be directly used in the basic level conceptualization. Unfortunately, when e is given, $P(c|e)$ is proportional to the co-occurrence of c and e. So it tends to map e to some general concepts. From another perspective, if we want to find in which concepts, e is its most typical instance, we can also try to ranked these concepts by $P(e|c)$.

However, the above naive formulas may cause some problems. E.g. there is a small concept called *microsoft's smaller and nimble rival*, which only contain one instance "apple" and their co-occurrence is 1. Then with this naive typicality formula, *P(apple | microsoft's smaller and nimble rival)* is 1. In this case, if we try to conceptualize "apple" with $P(e|c)$, this kind of small concepts will always rank highest.

To mitigate this issue, we propose a **Smoothed Typicality** for BLC:

$$P(c|e) = \frac{n(c,e)+\varepsilon}{\sum_{e \in c_i} n(c_i,e) + \varepsilon N_{concept}} \tag{6}$$

where $N_{concept}$ is the total number of concepts, and ε is a small constant which assumes every (concept, instance) pair has a very small co-occurrence probability in the real world, no matter whether we find some evidence from Web documents.

This smoothed typicality can achieve good results with a fine tuned ε. Details are discussed in the Experiment subsection 5.1.

4.2.3 BLC Using Representativeness Score

From another perspective, if we don't want to introduce ε into the basic level conceptualization, we can try to combine $P(c|e)$ and $P(e|c)$ in some way.

Intuitively, we define the **Representativeness** score for BLC as follows:

$$Rep(e,c) = P(c|e)P(e|c) \tag{7}$$

Then, given an instance e, we use the above formula to find its most representative concepts:

$$C_{rep}(e) = \max_{c} Rep(e,c) \tag{8}$$

where c is any concept that have an edge connecting to the given instance e in the semantic network. Conceptually, this function tries to boost up this kind of concepts: *given the instance, the concept is its typical concept; while given this concept, the instance is also its typical instance.*

Though Equation 7 is straightforward, we find it is quite useful for BLC in practice. Therefore, we try to analyze its essence, and find it is related to PMI and $commute\ time$. Their relations and differences will be discuss in detail in the following subsections.

4.2.4 Comparison with PMI

We observe that, with derivation, Formula 7 can be changed to:

$$Rep(e,c) = \frac{P(e,c)}{P(e)P(c)} * P(e,c) \tag{9}$$

The above fomula is similar with Pointwise mutual information (PMI) [29]. PMI is a standard measure of association in information theory. If we use PMI to calculate the representativeness between concept c and instance e, then we can get:

$$\begin{aligned} PMI(e,c) &= \log \frac{P(e,c)}{P(e)P(c)} \\ &= \log P(e|c) - \log P(e) \end{aligned} \tag{10}$$

However, since e is given, $\log P(e)$ is a constant. Then ranking by PMI is reduced to ranking by typicality $P(e|c)$ in our scenario. As we mentioned in subsection 4.2.1, this may cause that top concepts are too specific.

Bouma [14] proposes a normalized pointwise mutual information (NPMI). If we use $NPMI$ to calculate the representativeness between concept c and instance e, then we can get:

$$NPMI(e,c) = \frac{PMI(e,c)}{-\log P(e,c)} = \frac{\log P(e|c) - \log P(e)}{-\log P(e,c)} \quad (11)$$

Nevertheless, NPMI has the following problems in our scenario:

- When $P(e,c)$ is large (tending to be 1), $P(e,c)$ dominates $NPMI$ because "$-logP(e,c)$" tends to be 0. This leads to top representative concepts too general.
- When $P(e,c)$ is small, $\frac{1}{-\log P(e,c)}$ does not change much when $P(e,c)$ changes, thus, PMI dominates $NPMI$ in this case. As mentioned above, this leads to top representative concepts too specific.

Though both PMI and $NPMI$ are not suitable for calculating the representativeness score, if we take the logarithm of Formula 7, we can deduce that:

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

Actually, the above equation is PMI^2, which is a typical case of PMI^k family proposed by Daille [19]. Therefore, the logarithm of representativeness can be treated as a type of normalized PMI.

4.2.5 Comparison with Commute Time

Alternatively, we can also consider this problem of finding representative concepts from the graph perspective.

As we mentioned in subsection 4.2.1, given the instance e, the representative concept c should be one of e's typical concepts, in other words, c should have "shortest distance" with e. Similarly, given this representative concept c, e should have "shortest distance" with c. Therefore, we can treat *the process of finding representative concepts* as *a process of finding concept nodes have shortest expected distance with e*, we can formalize this process to a random walk problem of finding top nearest concepts reached by a given instance e, and leverage the commute time as the distance measure. Commute time [28] is a common measure of the distance between two nodes in a graph. It is the expected number of steps that a random walk starting at node i, go through node j once, and return to i again.

Then, we define the random walk should alternate between concepts and instance in the network. For a given instance e and a concept c, their commute time CT is:

$$\begin{aligned} CT(e,c) &= \lim_{T\to\infty} \sum_{k=1}^{T} (2k) * P_k(e,c) \\ &= \sum_{k=1}^{T} (2k) * P_k(e,c) + \sum_{k=T+1}^{\infty} (2k) * P_k(e,c) \\ &\geq \sum_{k=1}^{T} (2k) * P_k(e,c) + 2(T+1) * (1 - \sum_{k=1}^{T} P_k(e,c)) \end{aligned} \quad (13)$$

where $P_k(e,c)$ is the probability of starting from e through c and return to e at step $2k$.

As we only consider top concepts (e.g. the concept with commute time less than M), we can neglect the concept with commute time larger than M in only T steps. For instance, we constrain the random walk within 4 steps (treat #steps>4 as 4 steps), then:

$$\begin{aligned} CT'(e,c) &= 2 * P(c|e)P(e|c) + 4 * (1 - P(c|e)P(e|c)) \\ &= 4 - 2 * P(c|e)P(e|c) \\ &= 4 - 2 * Rep(e,c) \end{aligned} \quad (14)$$

According to the above derivation, our representativeness score has an inverse relationship with commute time under this random walk assumption. But for the complete commute time, our experiments show that the representativeness score is better than it.

4.3 Vagueness

For concepts in the semantic network, we observe that some of them are concrete (such as *country*, *city*, and *celebrity*), but some of them are vague (such as *thing*, *item*, and *factor*). By looking into vague concepts, we find they have the following characteristics:

- They contain many instances.
- Their instances are diverse.

Vague concepts are not very meaningful to differentiate the sense of a term, because almost every term can map to vague concepts. Therefore, we need to associate a **vagueness** score for each concept, and then ignore vague concepts in some application scenarios.

According to our observations, we derive the vagueness score of concept c from its instance diversity:

$$Vag(c) = \frac{\sum_{x,y\in c, x\neq y} D(x,y)}{|c| * (|c|-1)}, \quad |c| > 1 \quad (15)$$

where x and y are instances belong to the concept c, $|c|$ is the number of instances contained by c, and $D(x,y)$ is the distance between x and y.

$D(x,y)$ can be obtained by many approaches. Some are based on a knowledgebase such as WordNet[7, 6], others are based on a corpus[17, 13]. Considering practicability, in this paper, we leverage instances' concept distributions to calculate their distance:

$$\begin{aligned} D(x,y) &= 1 - cosine(CV(x), CV(y)) \\ &= 1 - \frac{\sum_{c_i=c_j} n(c_i,x) * n(c_j,y)}{\sqrt{\sum_{x\in c_i} n^2(c_i,x)} * \sqrt{\sum_{y\in c_j} n^2(c_j,y)}} \end{aligned} \quad (16)$$

where $CV(x)$ is the concept vector of the instance x, and $n(c_i,x)$ is the co-occurrence of concept c_i and instance x.

With this measure, top vague concepts are {entry, 0.956}, {option, 0.949}, {thing, 0.939}, {item, 0.904}. These concepts contain various entities. In contrast, concrete concepts have small scores, such as {country, 0.052}, {city, 0.118}, {celebrity, 0.280}.

4.4 Ambiguity

For instances in the semantic network, we observe that some instances are ambiguous, which means they have multiple senses. For example, the term "china" has senses *country* and *fragile item*; "apple" has senses *fruit* and *company*; "harry potter" has senses such as *book*, *movie*, and *character*. To identify these terms, we need an **ambiguity** level for all instances.

In Section 3, we clustered millions of concepts to thousands of concept clusters, and built hierarchies among them. These clusters and hierarchies can be treated as a kind of senses, and benefit the ambiguity detection. In this paper, we define the *sense* as *a hierarchy of concept clusters*.

For each instance, we first identify its senses, and then classify it to the 3 categories. We call them *Naive Ambiguity Levels*.

DEFINITION 4.3 (NAIVE AMBIGUITY). *Given an instance e, we classify its ambiguity to a following level:*

- L_0 ***(unambiguous)****: e contains only 1 sense. E.g. "apple juice" only has the sense related to* juice*.*
- L_1 ***(unambiguous and ambiguous both make sense)****: e contains 2 or more senses, but these senses are related. E.g. "Google" has senses like* company *and* search engine*, but these two senses are related.*
- L_2 ***(ambiguous)****: e contains 2 or more senses, and these senses are very different from each other. E.g. "python" has senses like* animal *and* programming language*, and these two senses are very different.*

To detect whether two senses s_i and s_j are related, we can compare every concept cluster pair in these two senses, and then select the maximum score as the similarity:

$$SenseSim(s_i, s_j) = \max\{ClusterSim(Cl_\alpha, Cl_\beta) \mid Cl_\alpha \in s_i \;\&\&\; Cl_\beta \in s_j\} \tag{17}$$

where Cl_α is a concept cluster in the sense s_i, Cl_β is a concept cluster in s_j, and $ClusterSim$ is the similarity function to calculate the similarity between clusters.

The cluster similarity can be obtained by comparing their entity distributions:

$$\begin{aligned} &ClusterSim(Cl_\alpha, Cl_\beta) = cosine(EV(\alpha), EV(\beta)) \\ &= \frac{\sum_{e_i=e_j} n(Cl_\alpha, e_i) * n(Cl_\beta, e_j)}{\sqrt{\sum_{e_i \in Cl_\alpha} n^2(Cl_\alpha, e_i)} * \sqrt{\sum_{e_j \in Cl_\beta} n^2(Cl_\beta, e_j)}} \end{aligned} \tag{18}$$

where $EV(\alpha)$ denotes the entity distribution of concept cluster Cl_α, $n(Cl_\alpha, e_i)$ is the total co-occurrence of concept cluster Cl_α and instance e_i: $n(Cl_\alpha, e_i) = \sum_{c \in Cl_\alpha} n(c, e_i)$.

In practice, we find that though some instances contain multiple senses, the weight of their first sense s_1 is much larger than their second sense s_2. E.g. the first sense of the instance "Taylor Swift" is related to *celebrity* (weight: 0.83), and its second sense is *song* (weight: 0.02). This kind of cases are caused by some rare knowledge facts or extraction errors. But actually, people tend to consider that "Taylor Swift" is unambiguous in common situation. Therefore, for the instance in L_2, we employ the following indicator to subdivide them:

$$\delta_e = \frac{\omega_{s_1} - \omega_{s_2}}{\omega_{s_2}} \tag{19}$$

where $e \in L_2$, ω_{s_1} is the weight of its first sense, and ω_{s_2} is the weight of its second sense. Then we propose *Refined Ambiguity Levels* as follows:

$$Amg(e) = \begin{cases} 0 & e \in L_0 \\ 1 & e \in L_1, or\ (e \in L_2 \;\&\&\; \delta_e > \alpha) \\ 2 & e \in L_2 \;\&\&\; 0 < \delta_e < \alpha \end{cases} \tag{20}$$

We will discuss the parameter tuning in the subsection 5.2.3.

4.5 Similarity

As we mentioned above, semantic networks are composed of terms. Measuring semantic similarity between terms (i.e., words or multi-word expressions) is a fundamental problem. It is critical to lots of applications. Intuitively, when we say two terms are semantically similar, it means their meanings are close, or they share many common properties. For example, "global company" and "multinational corporation" are similar because they have many similar instances. Another example, "ibm" and "microsoft" are similar because they share many similar concepts. A more complicated case is "apple" which has multiple senses. However, when people mention it with "microsoft", people will do sense disambiguation first, then compare them together and think they are very similar. We need to handle all of these cases while measuring term similarity.

It is natural that we can leverage the context information of terms in the semantic networks to measure their semantic similarity. The straightforward way to define similarity is

$$Sim(t_1, t_2) = cosine(R(t_1), R(t_2)) \tag{21}$$

where $R(t)$ is the vector representation of t's context in the semantic network. The context of a term depends on its type. If t is a concept, i.e., it appears more as a hypernym than a hyponym, then

$$R(c) = \langle p(e_1|c), \ldots, p(e_k|c) \rangle.$$

Else if t is an instance, i.e., it appears more as a hyponym, then

$$R(e) = \langle p(c_1|e), \ldots, p(c_k|e) \rangle.$$

Here $p(e|c)$ and $p(c|e)$ are typicality scores defined in subsection 4.1.

This basic approach works reasonably well for most occasions, but for those ambiguous terms ($Amg(e)$=2), their scores are affected by their multiple senses. We therefore propose the following refined approach for ambiguous instances comparison:

$$Sim(t_1, t_2) = \max_{x \in R_{cl}(t_1), y \in R_{cl}(t_2)} \{ClusterSim(x, y)\} \tag{22}$$

where $R_{cl}(t)$ is the set of t's concept clusters (by removing those clusters with vague concepts measured by Equation 15), and $ClusterSim(x, y)$ is defined by Equation 18.

5. EXPERIMENTS

To evaluate the effectiveness of scores proposed in this paper, we conduct our experiments on the Probase[3] [42] semantic network. The dataset we use is called "core" version. It contains 3,024,814 unique concepts, 6,768,623 unique instances, and 29,625,920 edges among them.

5.1 Evaluation on Typicality and Representativeness

5.1.1 Experiment Setting

Conceptualization is one of the essential processes for text understanding. It maps instances in the text to the concept space. Both "typicality" and "representativeness" scores can be used in the conceptualization. In this evaluation, we select 26 instances from top search queries between July 1, 2012 and December 31, 2012, then we use the typicality $P(c|e)$, $P(e|c)$, and representativeness $Rep(e, c)$ to rank concepts corresponding to these instances.

We also compare our approaches with several baselines: MI, $NPMI$, PMI^3. Since PMI is reduced to typicality $P(e|c)$ in our scenario, we don't treat it as another baseline separately. The formulas of these baselines are as follows:

Mutual Information(MI): $MI(e, c) = \sum P(e, c) \log \frac{P(e,c)}{P(e)P(c)}$

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

Normalized Pointwise Mutual Information(NPMI): please refer to Equation 11.

PMI^3 [14]: $PMI^3(e,c) = \log \frac{P(e,c)^3}{P(e)P(c)}$

For each approach, we try both cases: without smoothing, and with smoothing (please refer to subsection 4.2.2).

5.1.2 Metrics

We now discuss how we evaluate the results of different approaches. As there is not ground-truth ranking for conceptualization results. We manually label these *(instance, concept)* pairs (there are all 1,683 pairs), and assign a label for each pair. The labeling guideline is shown in Table 2.

Label	Meaning	Examples
Excellent	Good matched concepts	(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)

Table 2: Labeling Guideline for Conceptualization

Then we employ $precision@K$ and $nDCG$ to evaluate the results of different approaches. The $precision@K$ is used for evaluating the **correctness** of conceptualization results, and $nDCG$ is used to measure the **ranking** of concepts.

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:

$$Precision@K = \frac{\sum_{i=1}^{K} rel_i}{K} \tag{23}$$

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 $nDCG$ for top-K results as follows:

$$nDCG_K = \frac{rel_1 + \sum_{i=2}^{K} \frac{rel_i}{\log i}}{ideal\_rel_1 + \sum_{i=2}^{K} \frac{ideal\_rel_i}{\log i}} \tag{24}$$

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.

5.1.3 NDCG & Precision

We show the comparison from two aspects: without smoothing, and with smoothing.

Without smoothing, as Fig. 4 shows, the representativeness score $Rep(e,c)$ proposed by this paper is much better in precision than others. $Rep(e,c)$ is the best for top-2, top-5, and top-10. PMI^3 is also good for top-1, but it is worse for other cases. So it is not stable. For the ranking of top concepts, $Rep(e,c)$ outperforms all other methods in all cases.

With smoothing, we try different values of ε. We observe that the smoothing technique has a significant effect on the typicality $P(e|c)$, as shown in Fig. 5. For $P(e|c)$, its optimized ε is *1e-4*. With this smoothing setting, the typicality $P(e|c)$ outperforms all other methods in both $precision$ and $nDCG$ when K is 2, 3, 5, 10, and 15. However, $Rep(e,c)$ wins the $precision$@1 when its smoothing ε=*1e-4*.

We have the following conclusions:

- The representativeness score $Rep(e,c)$ performs best for conceptualization task overall, and it's robust.
- The typicality score $P(e|c)$ has a good performance after a sophisticated tuning on the smoothing value ε.
- The approach selection depends on: (1) the application scenario, e.g. the application focuses on the top-1 concept, or a list of good concepts; (2) existing resources, e.g. whether there is a labeled set for the smoothing value ε tuning.

5.1.4 Examples

Some examples are shown in Table 3. We find top concepts ranked by $P(c|e)$ tend to be general. E.g. the top 1 concepts of "jewelry," "car," and "battery" are all *item*. Obviously, *item* is not a good concept to predict these instances' features. On the other hand, concepts ranked by $P(e|c)$ (without smoothing) tends to be specific. Though these concepts have greater power on prediction, they have limited coverage. This makes them useless in the computation. E.g. when we compare two related instances with extremely specific concepts, they may have little overlapping information. In Table 3, concepts ranked by $Rep(e,c)$ and $P(e|c)$ with smoothing(ε = *1e-4*) are the best. They are the trade-off between general concepts and specific concepts.

5.2 Evaluation on Vagueness and Ambiguity

5.2.1 Setting & Metrics

Because there is no standard benchmark for vagueness and ambiguity evaluation, we still leverage top search queries between July 1, 2012 and December 31, 2012. We make search queries join concepts and instances in our semantic network:

- For vagueness evaluation, we select top 100 concepts order by the query frequency as per the test set.
- For ambiguity evaluation, we select the top 100 instances ordered by query frequency for each ambiguity level. Therefore, there are total 300 instances in the test set.

Then we manually label test sets with the same guideline, as shown in Table 4.

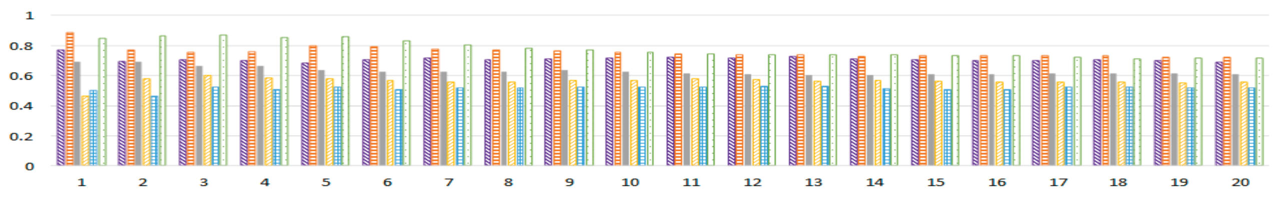
Label	Meaning	Examples
Yes	Vague concept or ambiguous instance	Concept: entity, item Instance: fox, apple
Borderline	Classified to both cases are fine	Concept: job, culture Instance: nike, facebook
No	Concrete concept or unambiguous instance	Concept: country, car Instance: computer, potato

Table 4: Labeling Guideline for Vagueness and Ambiguity

To evaluate the vagueness score of concepts, and ambiguity level of instances, we still employ $precision$ as the metrics.

5.2.2 Vagueness

To map our vagueness score to the labels, we do following transformations: (1) set a threshold α, treat the concept with score $Vag(c)$ $>\alpha$ as a vague concept; (2) treat the concept with human label "Yes" or "Borderline" as a vague concept. Then we compare the machine labeled set and human labeled set with different threshold α. The result of precision is shown in Fig. 6(a). As we can see, with α = 0.7, the precision is the highest: 0.78. Then we analyze the error cases. We find that most of errors come from concepts containing lots of named entities, such as *book, word, hotel, restaurant*. For these concepts, their vagueness score is high, because their instances are very diverse. From this perspective, our


(a) Precision Without Smoothing

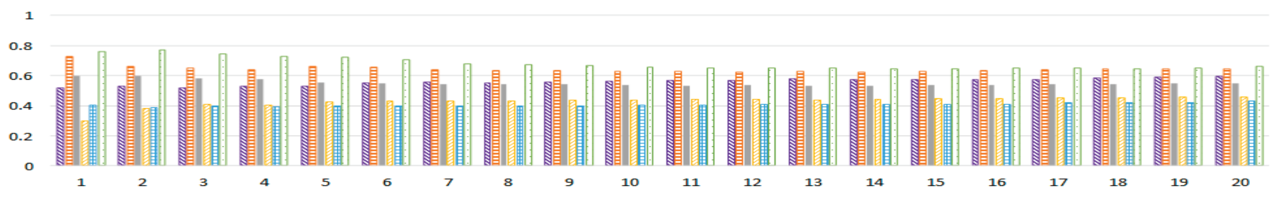
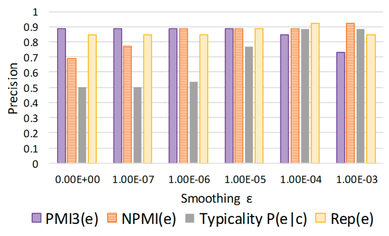
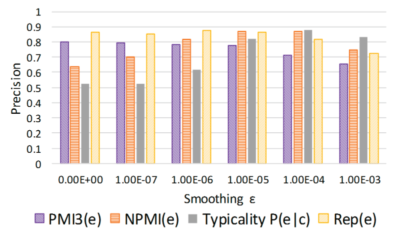
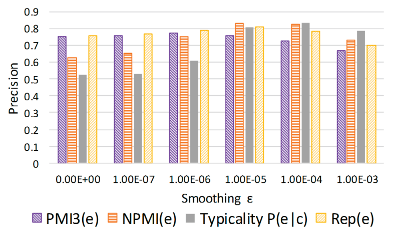
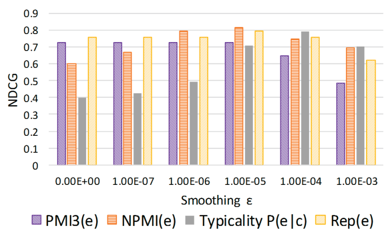

(b) nDCG Without Smoothing

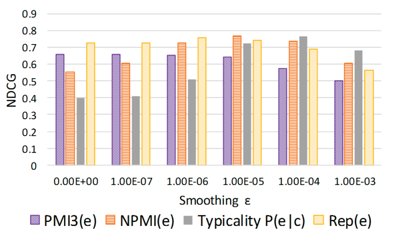
Figure 4: Precision and nDCG Comparison without Smoothing


(a) Precision@1


(b) Precision@5


(c) Precision@10


(d) nDCG@1


(e) nDCG@5

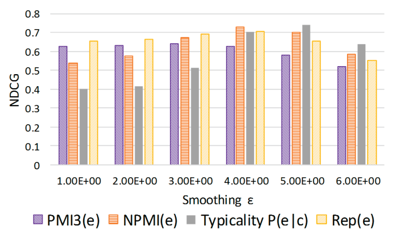
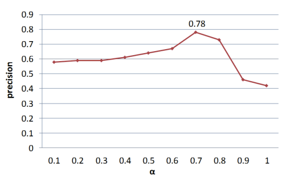

(f) nDCG@10

Figure 5: Precision and nDCG Comparison with Different Smoothing Values


(a) Precision of Vagueness

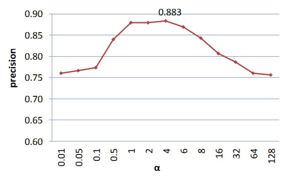

(b) Precision of Ambiguity

Figure 6: Precision of Vagueness and Ambiguity

vagueness score still makes sense for these concepts. By predefining a concept list for named entities (it is quite simple to discover this kind of concepts by checking whether there are lots of instances starting with a capital letter), our precision reaches up to 96%.

5.2.3 *Ambiguity*

For ambiguity, our scoring function (please refer to Equation 20) gives 3 scores: 0, 1, and 2, while the human labeled results are “Yes,” “Borderline,” and “No.” To align them, we obey following rules: (1) if $Amg(e)$ is 0 or 1, the result is correct when human label is “No” or “Borderline;” (2) if $Amg(e)$ is 2, the result is correct when human label is “Yes” or “Borderline.”

Then we get the precision curve of ambiguity with regard to the threshold α in Equation 20, as shown in Fig. 6(b). X-axis represents the value of α, and Y-axis represents the precision. From the results, we find the precision is high when α is between 1 and 6. The highest precision is 0.883 when α=*4*. We analyze the errors,

Table 3: Examples of Typicality and Representativeness Scores

Instance	Rank by $P(c\|e)$	Rank by $P(e\|c)$ (same as PMI in our scenario)	Rank by $Rep(e,c)$	Rank by $P(e\|c)$ with smoothing (ε = *1e-4*)
jewelry	item valuable accessory	typically female asset classic bridesmaid gift small glass piece	valuable valuable item accessory	valuable valuable item metal object
Nokia	company brand manufacturer	leading global MNC handset brand established technology vendor brand-name cell phone product	original mobile phone mobile phone manufacturer handset maker	mobile phone manufacturer handset maker handset manufacturer
Bluetooth	feature technology wireless technology	connectivity facility disable networking capability hands-free communication device	wireless technology wireless protocol connectivity option	wireless technology connectivity option wireless protocol
car	item vehicle asset	secured purchase big assest infrequent purchase	purchase vehicle noncash payment	purchase large purchase motor vehicle
battery	item accessory power source	mobile power source electrical energy storage system auto supply	power source power supply energy storage device	power source power supply consumable part

and find most of errors are caused by named entities with different roles. E.g. for the term "Arnold Schwarzenegger," its top senses are related to *actor*, *bodybuilder*, and *politician*. Since these senses have low overlapping of their instances, their pairwise sense similarities are low. Therefore, its $Amg(e)$ is 2 in our function. However, this kind of problems has been well studies, and known as named entities resolution or disambiguation [34, 15]. By leveraging their work, the precision of ambiguity can be further improved.

5.3 Evaluation on Term Similarity

5.3.1 Dataset & Metrics

Our evaluation data set consists of three parts: (1) M&C data set (28 pairs), which is a subset of Rubenstein-Goodenough's [37]; (2) WordSim203 (203 pairs), which is a subset from WordSim353 [1]; (3) W&P data set (300 pairs) [2], which is a human labeled set for semantic similarity between terms.

We compare our approach with other representative approaches. Hungarian[3] is a string-based approach, Troy [4] is string-based plus knowledge from WordNet, Sánchez [38] leverages information content and WordNet, and Bollegala [13] is based on search snippets.

To evaluate the effectiveness, we take Pearson Correlation Coefficient (PCC in short) as the metrics. It is a measure of the strength and direction of the linear relationship between the machine ratings (as X) and the human ratings (as Y) over our data set:

$$\rho = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}}$$

5.3.2 Results

The results are shown in Table 5. Because some approaches heavily rely on the WordNet, and WordNet doesn't cover all terms in our data set. Therefore, in the table, we divide the results into two parts: *terms in WordNet*, and *terms not in WordNet*. For both cases, our approach outperforms other competing methods. We analyze the results and make following observations: (1) string based approaches depend on surface forms of terms, and are not very suitable for semantic similarity; (2) WordNet based approaches are limited by its coverage; (3) snippet-based approaches tend to produce relatedness rather than similarity, because they leverage the co-occurrence of two terms instead of isA relations. Examples with our semantic similarity scores are shown in Table 6.

Table 5: Pearson Correlation Coefficient on Terms

Method	terms in WordNet	terms not in WordNet	All terms
Hungarian[3]	0.037	0.429	0.054
Troy[4]	0.531	0.325	0.468
Sánchez[38]	0.585	-	-
Bollegala[13]	0.521	0.511	0.505
Our approach	**0.761**	**0.665**	**0.735**

Table 6: Examples of Human Ratings and Similarity Scores

Pair	Human Rating	Similarity Score
$\langle lunch, dinner\rangle$	0.950	0.9987
$\langle notebook, laptop\ computer\rangle$	0.950	0.8538
$\langle global\ company, multinational\ company\rangle$	0.950	0.746
$\langle technology\ company, microsoft\rangle$	0.85	0.8208
$\langle high\ impact\ sport, competitive\ sport\rangle$	0.85	0.8155
$\langle employer, large\ corporation\rangle$	0.6	0.5353
$\langle travel, meal\rangle$	0.25	0.0426
$\langle music, lunch\rangle$	0.1	0.0116
$\langle company, table\ tennis\rangle$	0.05	0.0003

5.4 Application

With the scores proposed in this paper, the semantic network becomes usable for machines. In this subsection, we showcase one application leveraging these scores. Definitely, more applications can be easily developed for different scenarios.

Sponsored search is one of the most successful business models for search engine companies. It matches user queries to relevant ads. In reality, each ad is associated with a list of keywords. Advertisers bid for keywords, and also specify matching options for these keywords. One option is *exact match* where an ad is displayed only when a user query is identical to one of the bid keywords associated with the ad. Another option is *smart match* which is based on semantic relevance. Exact match targets exact traffic, but it is limited since queries are various. Currently, smart match is the default option provided by mainstream search engines.

In smart match, search engines map the query to bid ad keywords. Since both are short texts, traditional bag-of-words approaches do not work well in this scenario. Therefore, we can leverage the semantic network for this task in smart match.

Given a short text, we leverage the scores of semantic network as follows (we omit the details due to lack of space):

1. Identify instances from the short text (query, or bid keyword), map it to concept space with the representativeness score $Rep(e, c)$, and get a concept vector V;
2. Remove vague concepts whose $Vag(c) > 0.7$ from V;
3. For those ambiguous instances whose $Amg(e) = 2$, leverage their context information to do sense disambiguation;
4. For instances with high similarity scores $Sim(e_i, e_j)$ in the short text, we use Bayesian inference to boost up their common concepts.

For each short text, we simply merge the concept vectors of different instances in the short text, and get a single concept vector as the representation of this short text. Then we can calculate the semantic similarity between queries and bid keywords by comparing their concept vectors (E.g. using cosine similarity function).

We conduct our experiments on real ads click log of Bing (from November 1, 2011 to November 30, 2011). The experiment process is as follows: first, we calculate the semantic similarity score of (query, bid keyword) pairs for each record in the log; Second, we divide all scores into 10 buckets; Third, we aggregate the click number and impression number in the same bucket.

The overall results are illustrated in Fig. 7(a). X-axis represents the bucket number (E.g. *bucket 1* means the similarity score is between 0 and 0.1, and so on). Y-axis is the general click-through rate (CTR) of this bucket, where $CTR = \frac{\#\ of\ clicks}{\#\ of\ impressions}$. From the figure, we observe that: (1) once our semantic similarity score is low, the CTR is low; (2) once our score is high, the CTR is high; (3) the highest CTR is about 3 times of the lowest CTR. This means our semantic similarity score can be a strong feature for the query and ads matching.

We further analyze our results by query frequencies. We separate all queries to 10 deciles in the order of frequency, and each decile has the same total volume of traffic. Generally speaking, decile 1 to 3 are head queries, decile 4 to 6 are torso queries, and decile 7 to 10 are tail queries. Usually, head queries can be covered by exact match. Therefore we mostly focus on torso queries and tail queries. Results are shown in Fig. 7(b) and Fig. 7(c). For both torso and tail queries, the correlation between our semantic similarity score and CTR is preserved. This is quite good because long queries usually are lack of click signals, and the semantics can fill this gap.

6. RELATED WORK

There are two major efforts on acquiring knowledge for machines. One is building fact-oriented knowledgebases, the other is building term-based semantic networks.

The fact-oriented knowledgebases focus on collecting as more as possible knowledge facts. For example, the Cyc project [26] integrates 25 years of efforts by lots of domain knowledge experts. Their release ResearchCyc 1.1 [5] claims it has 5 million assertions. To overcome the bottleneck of contributors' limit, some knowledgebases such as Freebase [12] leverage community efforts to increase the scale. One of its release versions (the version of freebase-rdf-2013-08-26-test.gz) claims it has 1.2 billion fact triples. On the other hand, automatic knowledgebases construction has also been extensively studied in the literature. The most notable work is WikiTaxonomy [36] and YAGO [41]. The release version of YAGO2s (in August 2013) claims it contains more than 120 million facts about entities. However, as we discussed before, knowledge facts without sophisticated probability are limited in lots of applications, and can answer only a small portion of queries.

The term-based semantic networks focus on extracting concepts, instances, and their relations from web pages by using natural language processing and information extraction techniques. The most typical work is KnowItAll [20], TextRunner [8], NELL [16], and Probase [42]. In these projects, KnowItAll [20] and TextRunner [8] propose confidence scores to serve the purpose of filtering incorrect isA pairs. Probase [42] also proposes plausibility and typicality scores for taxonomy inference. However, all of these scores are specific for some ad-hoc purposes. In this paper, we want to propose a generic framework for scoring in common semantic networks. The scores described in this paper are basics for many kinds of knowledge empowered applications.

In terms of scores discussed in this paper, some of them are touched more or less in previous work.

Typicality and representativeness are actively studied in cognitive science and psychology at first. E.g. psychologist Gregory Murphy's highly acclaimed book [33] discusses the typicality and the basic level of concepts from the perspective of psychologists, which is the basis of these two scores proposed in this paper. Other work [25, 42] proposes typicality scores for some special scenarios. Their scoring functions cannot be easily extended to semantic networks. E.g. Lee et al. [25] leverage instances as intermedia for calculating typicality $P(a|c)$ between an attribute a and a concept c. Instead, our typicality is more generic and easy to be adopted by more applications. Besides, we also propose a smoothing approach and representativeness score, which has been proved that they are much better than the standard typicality in some scenarios.

Ambiguity for words are widely studies in previous work [35]. It's also well known as word sense disambiguation(WSD). WordNet [21] is usually used in WSD. However, for those multi-word expressions, previous work cannot resolve them. Song et al. [39] try to identify ambiguous queries by using a supervised learning approach based on search results from the web. In this paper, we define the ambiguity for terms (including words and multi-word expressions). Then we leverage senses in the semantic network to measure a term's ambiguity level.

Existing efforts on similarity between terms mainly follow two ways: one is based on knowledgebases such as WordNet [4, 38], the other is based on terms' context from text corpora such as search snippets and web documents [13]. Our approach for terms' similarity belongs to the former way. Compared with existing work, our similarity measure function is more sophisticated, and leverages concept clustering and typicality proposed in this paper.

7. CONCLUSION

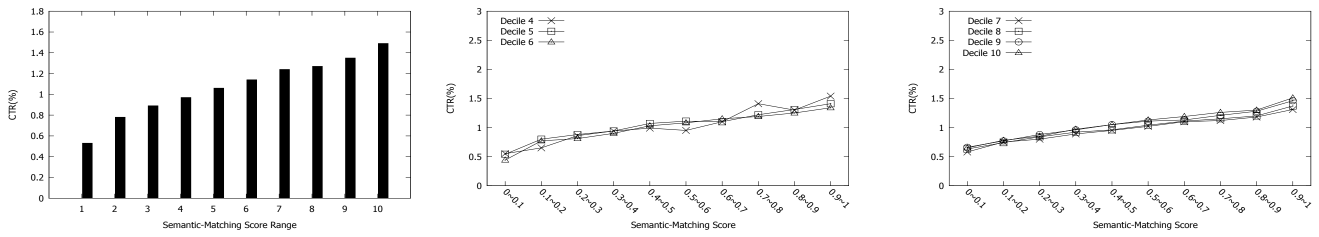
In this paper, we propose five fundamental scores *Typicality, BLC (Basic level conceptualization), Vagueness, Ambiguity,* and *Similarity* in semantic networks. We make deep analysis of these scores, and carefully design their formulas. With these scores, semantic networks become usable for machines in the text understanding and other key applications. We conduct extensive experiments and show the effectiveness of these scores. We also use a real example to demonstrate how these scores help improve current ads system.

8. REFERENCES

[1] http://www.cs.technion.ac.il/~gabr/resources/data/wordsim353/.

[2] http://adapt.seiee.sjtu.edu.cn/similarity/SimCompleteResults.pdf.

[3] http://www.math.uwo.ca/~mdawes/courses/344/kuhn-munkres.html.



(a) Different SMS Range (b) Torso Queries (c) Tail Queries

Figure 7: Correlation between Semantic Matching Score and CTR

[4] `http://www.codeproject.com/KB/string/semanticsimilaritywordnet.aspx`.

[5] Researchcyc. `http://www.cyc.com/platform/researchcyc`.

[6] E. Agirre, M. Cuadros, G. Rigau, and A. Soroa. Exploring knowledge bases for similarity. In *Proceedings of LREC'10*.

[7] M. Alvarez and S. Lim. A graph modeling of semantic similarity between words. In *ICSC*, 2007.

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

[9] L. W. Barsalou. Ideals, central tendency, and frequency of instantiation as determinants of graded structure in categories. *JEP:LMC*, 11(4):629–654, 1985.

[10] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *JMLR*, 3:993–1022, 2003.

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

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

[13] D. Bollegala, Y. Matsuo, and M. Ishizuka. A web search engine-based approach to measure semantic similarity between words. *TKDE*, 23:977–990, 2011.

[14] G. Bouma. Normalized (pointwise) mutual information in collocation extraction. In *GSCL*, 2009.

[15] R. C. Bunescu and M. Pasca. Using encyclopedic knowledge for named entity disambiguation. In *EACL*, 2006.

[16] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. H. Jr., and T. M. Mitchell. Toward an architecture for never-ending language learning. In *AAAI*, pages 1306–1313, 2010.

[17] H. Chen, M. Lin, and Y. Wei. Novel association measures using web search with double checking. In *Proceedings of the COLING/ACL 2006*, pages 1009–1016, 2006.

[18] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *JMLR*, 12:2493–2537, 2011.

[19] B. Daille. *Approche mixte pour l'extraction de terminologie: statistique lexicale et filtres linguistiques*. PhD thesis, 1994.

[20] O. Etzioni, M. Cafarella, and D. Downey. Webscale information extraction in knowitall (preliminary results). In *WWW*, 2004.

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

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

[23] T. Hofmann. Probabilistic latent semantic indexing. In *SIGIR*, 1999.

[24] D. Kim, H. Wang, and A. Oh. Context-dependent conceptualization. In *IJCAI*, 2013.

[25] T. Lee, Z. Wang, H. Wang, and S.-w. Hwang. Attribute extraction and scoring: A probabilistic approach. In *ICDE*, 2013.

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

[27] P. Li, H. Wang, K. Zhu, Z. Wang, and X. Wu. Computing term similarity by large probabilistic isa knowledge. In *CIKM*, 2013.

[28] L. Lovász. Random walks on graphs: A survey. *Combinatorics, Paul erdos is eighty*, 2(1):1–46, 1993.

[29] C. D. Manning and H. Schütze. *Foundations of statistical natural language processing*, volume 999. MIT Press, 1999.

[30] C. B. Mervis, J. Catlin, and E. Rosch. Relationships among goodness-of-example, category norms, and word frequency. *Bulletin of the Psychonomic Society*, pages 283–284, 1976.

[31] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In *Proceedings of Workshop at ICLR*, 2013.

[32] A. W. Moore. An intoductory tutorial on kd-trees. Technical report, 1991.

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

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

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

[36] S. P. Ponzetto and M. Strube. Deriving a large-scale taxonomy from wikipedia. In *AAAI*, pages 1440–1445, 2007.

[37] H. Rubenstein and J. B. Goodenough. Contextual correlates of synonymy. *CACM*, 8(10):627–633, 1965.

[38] D. Sánchez, M. Batet, and D. Isern. Ontology-based information content computation. *Knowledge-Based Systems*, 24:297–303, 2011.

[39] R. Song, Z. Luo, J.-Y. Nie, Y. Yu, and H.-W. Hon. Identification of ambiguous queries in web search. *IPM*, 45:216–229, 2009.

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

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

[42] W. Wu, H. Li, H. Wang, and K. Q. Zhu. Probase: A probabilistic taxonomy for text understanding. In *SIGMOD*, pages 481–492. ACM, 2012.