Computing Term Similarity by Knowledge from Big Data

Peipei Li #*1, Haixun Wang *2, Kenny Q. Zhu †3, Zhongyuan Wang *2, Xindong Wu #4

# *Hefei University of Technology*
1 v-pli@microsoft.com
4 xwu@uvm.edu

* *Microsoft Research Asia*
2 {haixunw,zhy.wang}@microsoft.com

† *Shanghai Jiao Tong University*
3 kzhu@cs.sjtu.edu.cn

ABSTRACT

Computing semantic similarity between two terms is essential for a variety of text analytics and understanding applications. However, existing approaches are more suitable for semantic similarity between words rather than the more general multi-word expressions (MWEs), and they do not scale very well. Therefore, we propose a lightweight and effective approach for semantic similarity using a large scale semantic network automatically acquired from billions of web documents. Given two terms, we map them into the concept space, and compare their similarity there. Furthermore, we introduce a clustering approach to orthogonalize the concept space in order to improve the accuracy of the similarity measure. Extensive studies demonstrate that our approach can accurately compute the semantic similarity between terms with MWEs and ambiguity, and significantly outperforms 12 competing methods.

1. INTRODUCTION

Measuring semantic similarity between terms is a fundamental problem in lexical semantics [11] and it finds many applications in web and document search, question and answer systems, and other text analytics and text understanding scenarios. By *terms*, we mean either single words or multi-word expressions (MWEs). We say two terms are semantically similar, if their meanings are close, or the concept or object that they represent share many common attributes. For example, "emerging markets" and "developing countries" are similar because their semantic contents (the subset of countries) are very similar. Another example, "Google" and "Microsoft" are similar because they are both software companies. However, "car" and "journey" are not semantically similar but *related* because "car" is a transport means for the activity "journey". Specifically, semantic similarity is defined by some measure of *distance* between two terms on an isA taxonomy. It is clear that "car" and "journey" are quite far away from each other in an isA taxonomy from WordNet as shown in Figure 1. Semantic similarity is a more specific relationship and is much harder to model than *relatedness* (which can be modeled by term co-occurrence).

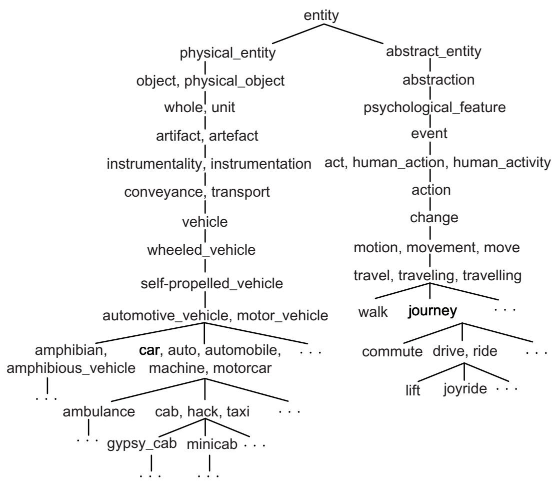

Figure 1: A fragment from WordNet showing semantic distance between "car" and "journey"

Recent work on term similarity can be roughly classified into two main categories: *knowledge based* and *corpus based*. Knowledge based approaches rely on handcrafted resources such as thesauri, taxonomies or encyclopedias, as the context of comparison. Most work in this space [20, 22, 5] depends on the semantic isA relations in WordNet [18] which is a manually curated lexicon and taxonomy. Corpus based approaches work by extracting the contexts of the terms from large corpora and then inducing the distributional properties of *words* or *n-grams*. Corpus can be anything from web pages, web search snippets to other text repositories.

One significant challenge faced by the knowledge-based methods is the limited coverage of taxonomies such as WordNet (with 155,287 words at last count). It does not cover many proper nouns (e.g., "Microsoft" or "Google"), or very popular senses (e.g., Apple the company or Jaguar the car make). Another major restriction of WordNet is that it primarily covers single words with only a handful of phrases or multi-word expressions. For example, it does not know "General Electric" or "emerging markets". For example, the similarity between "General Electric" and "GE" completely fail even though they are exactly the same thing.

Corpus-based approaches also face several serious limitations. First, such measures are biased because of the indexing and ranking mechanisms used in search engines. For example when querying the term "date" or "range" on Google, none of the first 100 results has anything to do with fruits (a sense for date) or cooking stoves (a sense for range), because these are rare senses of the two terms. With such search results, it is not surprising that a corpus-base method would think "asian pear" and "date" share very little commonality. Second, some search-result oriented similarity methods require interaction with the search engine which has high communication overhead and high index costs, and are not suitable for online applications. Third, statistical distribution based on words or n-grams in the context ignores the fact that i) the semantic units can be MWEs and not words, let alone n-grams; and ii) many words or phrases are ambiguous in meaning. Finally, corpus-based methods focus on surrounding context of a term or the co-occurrence of two terms within a neighborhood, both of which are more suitable to the calculation of semantic relatedness rather than similarity. Under this approach, "car" and "journey" would have high semantic relatedness because they co-occur very frequently on web texts.

In this paper, we propose a light-weight but effective framework for computing semantic similarity between a pair of terms using a large scale, general purpose isA network obtained from a web corpus. Below is a small sample of results:

- High similarity (synonyms): $\langle general\ electric, ge\rangle$
- High similarity (ambiguous terms): $\langle microsoft, apple\rangle$, $\langle orange, red\rangle$
- Low similarity (though share same hypernyms in WordNet): $\langle music, lunch\rangle$, $\langle banana, beef\rangle$
- Low similarity (related but not similar): $\langle apple, ipad\rangle$, $\langle car, journey\rangle$

The main contributions of this paper are:

- *Our approach has better coverage.* The semantic network behind this approach is one order of magnitude larger than WordNet in terms of the number of hypernym-hyponym relations. Our approach computes similarity between almost any two known noun-based MWEs.
- *Our approach produces more meaningful similarity.* Unlike corpus-based methods which can confuse similarity with relatedness, this approach calculates similarity by relations induced from an isA semantic network. It also seeks to disambiguate terms and thus excludes noises from irrelevant senses from the probability distributions.
- *Our approach is lightweight.* The most expensive clustering algorithm is performed offline. The remaining similarity function can be efficiently computed online. On average, it takes merely 65 milliseconds to compute the similarity for a pair of terms.

The rest of the paper is organized as follows. Section 2 introduces the preliminaries of Probase, our isA semantic network. Section 3 describes a basic algorithm for computing term similarity using Probase. Section 4 proposes an important refinement to the basic algorithm which addresses several key challenges faced by the basic approach. Section 5 gives some experimental results that compare our approach with a whole list of other previous approaches both using knowledge and using external corpora. Finally we discuss some related work in Section 6 and conclude in Section 7.

2. SEMANTIC NETWORK AND SYNSETS

To compute the similarity between two terms, we compute the similarity between their contexts. The context that we use in this paper comes from a large-scale, probabilistic semantic network, known as Probase [26]. Besides other knowledge, Probase contains isA relations between concepts, sub-concepts, and entities, which is called Γ_{isA} in this paper. For example, "Microsoft is a company". Here "company" is a *concept* and "Microsoft" is an *entity*. We refer to concepts and entities collectively as terms in this paper. Probase has the following important properties:

- Probase introduces a very large concept space with over 2.7 million concepts;
- It is not a tree structured taxonomy, but a network: An entity or concept may have many super-concepts. The benefit is that such links are data driven rather than handcrafted.
- Each isA relation (e isA c) is associated with conditional probabilities $P(e|c)$ and $P(c|e)$ (a.k.a. typicality scores).

Before we deal with similarity between any two terms, we first look at terms that have the same meaning. Intuitively, they should have the highest similarity. A single term may have many surface forms:

synonyms: "GE" and "General Electric"; "corporation", "firm", and "company";

spelling styles: "2d barcode" vs. "2d bar code" and "accomplished artist" vs. "accomplished artiste';

singular/plural forms: "shoe" vs. "shoes";

We address this issue in two steps. First, we use sources such as Wikipedia Redirects, Wikipedia Internal Links, and synonym data set in WordNet to group terms that are synonyms. Second, we use the edit distance function to evaluate the distance between terms as follows.

$$dis_{lex}(t_1, t_2) = \frac{EditDistance(t_1, t_2)}{MaxLength(t_1, t_2)}$$

If $dis_{lex}(t_1, t_2) < \varphi$, the two terms in the current pair are ones with very similar surface forms, and we group them together. In this paper, we set φ to 0.05 according to empirics, which enables high accuracy (95%) for identifying synonymous pairs.

At this point, all lexically similar or synonymous terms are grouped into a cluster which is analogous to the notion of "synset" in WordNet. As a result, the isA pairs between terms are also mapped logically into isA relations between synsets. The set of all synsets is called Γ_{ssyn} which provides a mapping between any Probase term, to its synset and hence all the other terms in that synset. When computing the semantic similarity between two terms which belong to the same synset, e.g., General Electric and GE, the similarity is set to the highest score, namely 1.

3. BASIC APPROACH

This section presents the basic framework of computing semantic similarity between two terms. In nutshell, given a pair of terms $\langle t_1, t_2 \rangle$, we first determine the type of the terms, i.e., whether they are concepts or entities, and then obtain the contexts of t_1 and t_2, i.e., $T(t_1)$ and $T(t_2)$, and finally compute the similarity between the two contexts.

$$\textsc{sim}(t_1, t_2) = sim(T(t_1), T(t_2)) \tag{1}$$

where $sim(c_1, c_2)$ is a similarity function for contexts.

3.1 Type Checking

Type checking requires the following data from the semantic network: 1) the entity and concept sets; 2) the isA relations between terms and their frequencies in corpus. If the given pair of terms has an isA relation, then the hypernym term is said to be a concept term while the hyponym term is an entity term. Otherwise, we decide the type of each term individually: t is a concept if its frequency as a concept is larger than its frequency as an entity; it is an entity otherwise.

3.2 Context Representation

We extract the context of a term according to its type and its position in the semantic network. If the term is a concept, its context is all the entities that it subsumes; if it is an entity, its context is all the concepts that it belongs to. Furthermore, we transform the context into a vector $\mathcal{I}_c$ or $\mathcal{I}_e$, where each element is the typicality score between the term and a term in the context:

$$\mathcal{I}_c = \langle w'_1, \cdots, w'_k \rangle \tag{2}$$

where $w'_i = p(e_i|c)$, $p(e_i|c)$ is the typicality of score for c and entity e_i, that is, how typical e_i is among all the entities c subsumes.

$$\mathcal{I}_e = \langle w_1, \cdots, w_k \rangle \tag{3}$$

where $w_i = p(c_i|e)$, and $p(c_i|e)$ is the typicality of score for e and concept c_i, that is, how typical c_i is among all the concepts e belongs to.

3.3 Context Similarity

We use the cosine similarity function [1] to evaluate the similarity between two contexts, i.e.,

$$sim(T(t_1), T(t_2)) = cosine(T(t_1), T(t_2)) \tag{4}$$

The complete algorithm for the basic approach is shown in Algorithm 1. We set top $K = 5$ by the empirical study.

3.4 Discussion

[1]Our experiments reveal that the cosine function outperforms other similarity/distance evaluation functions, such as Jaccard, JaccardExtended, Jensen-Shannon and the smoothed KL divergence.

Algorithm 1 Basic Approach

```
Input: $\langle t_1, t_2 \rangle$: a pair of terms;
       $\Gamma_{isA}$: the semantic network of isA relationship;
       $\Gamma_{ssyn}$: the synset data set in $\Gamma_{isA}$;
Output: a similarity score of $\langle t_1, t_2 \rangle$;
 1: if $t_1$ and $t_2$ belong to the same synset according to $\Gamma_{ssyn}$ then
 2:     Let $sim(t_1, t_2) \leftarrow 1$ and return $sim(t_1, t_2)$;
 3: end if
 4: Judge the type for each term;
 5: if $\langle t_1, t_2 \rangle$ is a concept pair then
 6:     Collect all entities of $t_i$ from $\Gamma_{isA}$ as the context and generate the entity vector $\mathcal{I}_c^{t_i} (i \in \{1, 2\})$ as defined in Eq. (2);
 7:     return $sim(\mathcal{I}_c^{t_1}, \mathcal{I}_c^{t_2})$ by comparing the context vectors $\mathcal{I}_c^{t_1}$ and $\mathcal{I}_c^{t_2}$ in Eq. (4);
 8: end if
 9: if $\langle t_1, t_2 \rangle$ is an entity pair then
10:     Collect all concepts of $t_i$ from $\Gamma_{isA}$ as the context and generate the concept vector $\mathcal{I}_e^{t_i} (i \in \{1, 2\})$ as defined in Eq. (3);
11:     return $sim(\mathcal{I}_e^{t_1}, \mathcal{I}_e^{t_2})$ by comparing the context vectors $\mathcal{I}_e^{t_1}$ and $\mathcal{I}_e^{t_2}$ in Eq. (4);
12: end if
13: if $\langle t_1, t_2 \rangle$ is a concept-entity pair then
14:     Collect top $K$ concepts of the entity term $t_i$ from $\Gamma_{isA}$ as the context $C^{t_i} (i \in \{1, 2\})$;
15:     for each concept $c_x$ in $C^{t_i}$ ($c_x \neq t_j$, $1 \leq x \leq K$) do
16:         $sim_{c_x} \leftarrow$ get the semantic similarity between $c_x$ and $t_j$ by repeating this algorithm iteratively;
17:     end for
18:     return $\max_{c_x \in C^{t_i}} \{sim_{c_x}\}$;
19: end if
```

Our preliminary evaluation shows that the basic approach works reasonably well for many pairs of terms, but for ambiguous terms with multiple senses such as *apple* and *orange*, the result is less satisfactory.

Table 1: Impact of Ambiguity on Similarity

Pair	Similarity Score
$\langle microsoft, google \rangle$	0.993
$\langle$**apple**$, pear \rangle$	0.916
$\langle$**apple**$, microsoft \rangle$	**0.378**
$\langle$**orange**$, red \rangle$	**0.491**

For example, as shown in Table 1, the basic approach decides that $\langle microsoft, google \rangle$ and $\langle apple, pear \rangle$ are quite similar whereas $\langle apple, microsoft \rangle$ and $\langle orange, red \rangle$ are not, because “apple” and “orange” have multiple senses. The dominant senses of “apple” and “orange” are a fruit, and we can see when we are comparing similarity using non-dominant senses, the results are less satisfactory.

4. REFINED APPROACH

The baseline approach introduced in Section 3 is not sensitive to different senses of a term. A simple solution is to use an existing knowledge database containing sense labels of terms such as the glosses in WordNet. But none of the handcrafted knowledge bases has the sufficient data coverage. Instead we propose the following refined approach.

Given a term, we define its *concept context* as the entire set of concepts that the term belongs to in Probase. We perform automatic sense disambiguation by concept clustering. We then prune irrelevant clusters as an optimization. Finally, we define the similarity of two terms as the highest similarity between any sense of the first term and any sense of the second term. Next we present this approach in details.

4.1 Concept Clustering

To identify multiple senses of a term automatically, we first use a k-Medoids clustering algorithm on the concept context of the term, and then we select the center concept in each cluster to represent a sense of this term. Figure 2(a) shows the concept context of the term "apple", and Figure 2(b) shows the clustered concepts. It is clear that each cluster represents a sense of the term.

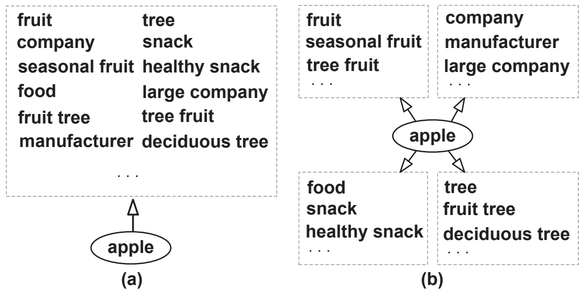

Figure 2: The concept context of "apple"

In the following, we define the distance measure and present the clustering algorithm.

4.1.1 Clustering Algorithm

We first define the semantic distance between two concepts c_1 and c_2 as

$$d_{sem}(c_1, c_2) = 1 - cosine(\mathcal{I}_{c_1}, \mathcal{I}_{c_2}) \tag{5}$$

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

Our algorithm is a modified k-Medoids clustering algorithm that partitions concepts according to their entity distributions. Good initial centers are essential for the success of partitioning clustering algorithms such as K-Medoids. Instead of using random initial centers, we identify good initial centers incrementally by a refined method from Moore [19]. The first medoid is randomly selected among all candidate points (concepts). Then we select the point that has the maximum of the minimum of the distances from each of the existing medoids to be the next medoid, i.e.,

$$m = \{m_i | \max_{c_j}\{\min_{i}\{d_{sem}(m_i, c_j)\}\} > \alpha\} \tag{6}$$

where c_j indicates the j^{th} point in the candidate points, m_i indicates the i^{th} medoid in existing medoids. This process continues until we do not find any medoids satisfying Eq. (6). In this case, we get k medoids at iteration 0: $M^0 = \{m_1^0, ..., m_k^0\}$. Clearly, the value of k is determined by the threshold α. The larger the threshold of α, the small the value of k. Since experiments show that the numbers of clusters do not vary much with α between 0.7 to 0.8, we set $\alpha = 0.7$ as an optimal value.

With k medoids in the t^{th} iteration, we assign each candidate concept $c_i \in C$ to its closest medoid $m^* \in M^t = \{m_1^t, ..., m_k^t\}$, namely, a medoid m^* with the minimum semantic distance from c_i:

$$m^* = \operatorname*{argmin}_{m_j^t \in M^t} d_{sem}(c_i, m_j^t) \tag{7}$$

When we assign all candidate concepts to the corresponding clusters, we can update the medoid with the most centrally located concept in each cluster. To find such a center concept, we first compute the average distance of a cluster C_i in terms of the semantic distance in Eq. (5) as

$$m_i^{t+1} = \operatorname*{argmin}_{c_y \in C_i} \left(\sum_{c_x \in C_i} \frac{d_{sem}(c_x, c_y)}{|C_i|} \right) \tag{8}$$

The clustering process iterates until the following objective function reaches minimum.

$$F(W, M) = \sum_{i=1}^{k} \sum_{j=1}^{n} w_{ij} d_{sem}(m_i, c_j) \tag{9}$$

where $w_{ij} \in \{0, 1\}$, $\sum_{i=1}^{k} w_{ij} = 1$, $0 < \sum_{j=1}^{n} w_{ij} < n$, k $(< n)$ is a known number of centers, n is the count of objects (concepts) to cluster. $W = [w_{ij}]$ is a $k \times n$ binary matrix, $M = [m_1, \ldots, m_k]$ is a set of cluster medoids and m_i is the i^{th} cluster medoid.

We use Eq. (8) to calculate the medoid set M. When M is computed, to minimize $F(W, M)$, W is given by

$$w_{ij} = \begin{cases} 1 & \text{if } d_{sem}(m_i, c_j) < d_{sem}(m_h, c_j) \\ & \quad (1 \le h \le k, \ h \ne i) \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

The convergence condition is that $F(W^t, M^{t+1}) - F(W^t, M^t)$ is less than a threshold δ (e.g., 10^{-5}). The minimization of F with the above constraints is an undecidable constrained nonlinear optimization problems. A partial optimization for M and W is shown in Algorithm 2.

```
Algorithm 2 Concept Clustering
Input: $C = \{c_1, ...c_j, ...\}$: the concept set;
       $\alpha$: the threshold in the limit of initial medoid count;
       $T$: the maximum iteration count;
       $\Gamma_{isA}$: the semantic network of isA relationship;
Output: $k$ clusters $\{C_1, ..., C_k\}$;
 1: Initialize the iteration time $t = 0$;
 2: Generate an initial medoid set $M^t = [m_1^t, m_2^t, \cdots, m_k^t]$
    incrementally by Eq. 6;
 3: Assign each concept $c_i$ to a cluster $C^*$ with a medoid
    $m^*$ satisfying Eq. (7);
 4: Update the weight matrix $W^t$ in Eq. (10) to make sure
    $F(W^t, M^t)$ is minimum;
 5: Update cluster medoids in $M^{t+1}$ with the most centrally
    located point in each cluster corresponding to Eq. (8);
 6: Calculate $F(W^t, M^{t+1})$ in Eq. (9);
 7: if $F(W^t, M^{t+1})$-$F(W^t, M^t) > \delta$ and $t < T$ then
 8:     Let $t = t+1$ and go to Step 3;
 9: end if
10: return clusters $\{C_1, ..., C_k\}$;
```

4.1.2 *Offline Concept Clustering*

The k-Medoids clustering algorithm has a time complexity of $O(kn^2)$, where k is the number of centers and n is the number of objects (concepts) to cluster. This is not acceptable if the number of pairs is large. To improve the efficiency, we cluster all concepts in the semantic network offline, and then during online calculation, each concept in a term's context can be quickly mapped to an offline cluster which acts as synset, and this effectively reduces the online clustering complexity to $O(n)$.

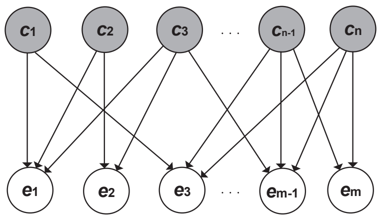

Figure 3: A concept-entity bipartite graph

To cluster the concepts in the semantic network, we use the entity distributions to represent the concepts and evaluate their similarities by Eq. (5). According to the isA relationships between concepts and entities in Γ_{isA}, we can construct a bipartite graph between concepts and entities (Figure 3) and cluster the concepts based on this graph. The basic idea is that if two concepts share many entities, they are similar to each other. From this bipartite graph, we represent each concept c_i as an L2-normalized vector as shown in Eq. (2), where each dimension corresponds to an entity in the graph.

Even though the number of concept and entity nodes may be large, the graph is actually very sparse. For example, a concept is connected with an average number of 5.72 entities in Γ_{isA}. Each entity is also connected to a couple of concepts on average. Therefore, for a concept c, the average size of S_c, the set of concepts which share at least one entity with c, is small. To find the closest cluster to c, we only need to check the clusters which contain at least one concept in S_c. Since each concept belongs to only one cluster in our method, the average number of clusters to be checked is small. Furthermore, edges in the graph with low weights (i.e., low typicality scores) are likely to be noises and can be ignored.

4.2 The Max-Max Similarity Function

In the basic approach, we compute the similarity of two terms by the cosine similarity between their contexts. In the refined approach, we use a new similarity function known as *max-max similarity* which is useful in identifying rare senses of terms with small sized clusters. Let $C = \{C_1, C_2, ..., C_k\}$ be clusters of all concepts in Γ_{isA}, C^{t_1} and C^{t_2} be the sets of concepts that two terms belong to respectively. According to the cluster information in C, we divide C^{t_1} and C^{t_2} into small clusters $C^{t_1} = \{C_1^{t_1}, ..., C_m^{t_1}\}$ and $C^{t_2} = \{C_1^{t_2}, ..., C_n^{t_2}\}$ by comparing $C^{t_1} \cap C$ and $C^{t_2} \cap C$ respectively. We then compute the similarity between the contexts of each cluster pair and get the semantic similarity between two terms as:

$$sim(T(t_1), T(t_2)) = \max_{x,y}\{cosine(C_x^{t_1}, C_y^{t_2})\} \tag{11}$$

where $1 \leq x \leq m$ and $1 \leq y \leq n$.

With all concepts clustered offline and the new similarity function based on concept clusters, the refined algorithm is given in Algorithm 3.

Algorithm 3 Refined Approach

```
Input: ⟨$t_1, t_2$⟩: a pair of terms;
       $\Gamma_{isA}$: the semantic network of isA relationship;
       $\Gamma_{ssyn}$: the synset data set in $\Gamma_{isA}$;
       $\Gamma_{cluster}$: clusters of all concepts in $\Gamma_{isA}$;
Output: a similarity score of ⟨$t_1, t_2$⟩;
 1: Install the synset checking and type checking as Steps
    1-4 in Algorithm 1;
 2: if ⟨$t_1, t_2$⟩ is a concept pair then
 3:     return $sim(\mathcal{I}_c^{t_1}, \mathcal{I}_c^{t_2})$ as Steps 6-7 in Algorithm 1;
 4: end if
 5: if ⟨$t_1, t_2$⟩ is an entity pair then
 6:     $sim_1 \leftarrow sim(\mathcal{I}_e^{t_1}, \mathcal{I}_e^{t_2})$ as Steps 10-11 in Algorithm 1;
 7:     Find clusters of contexts $C^{t_1}$ and $C^{t_2}$ from $\Gamma_{cluster}$;
 8:     $sim_2 \leftarrow sim(C^{t_1}, C^{t_2})$ computed in Eq. (11);
 9:     return $max(sim_1, sim_2)$;
10: end if
11: if ⟨$t_1, t_2$⟩ is a concept-entity pair then
12:     Collect all concepts of the entity term $t_i$ from $\Gamma_{isA}$ as
        the context $C^{t_i} (i \in \{1, 2\})$;
13:     Find clusters of contexts $C^{t_i}$ from $\Gamma_{cluster}$;
14:     for each cluster $C_x$ in $C^{t_i}$ do
15:         Select top $K$ concepts to represent $t_i$, namely $C_x^K =$
            $\{c_y | c_y \neq t_j, c_y \in C_x, 1 \leq y \leq K\}$;
16:         for each concept $c_y$ in $C_x^K$ do
17:             $sim_{c_y} \leftarrow$ get the semantic similarity between $c_y$
                and $t_j$ by repeating this algorithm iteratively;
18:         end for
19:         $sim_{C_x} \leftarrow max_{c_y \in C_x^K}\{sim_{c_y}\}$;
20:     end for
21:     return $max\{sim_{C_x} | C_x \in C^{t_i}\}$;
22: end if
```

4.3 Optimization by Cluster Pruning

The cluster-based refined approach improves the quality of similarity remarkably from the basic algorithm. But there are two problems. First, the max-max similarity function tends to boost the probability of picking a less dominant sense of a term because it is easier for small clusters to look similar by the cosine similarity and hence dominate the max-max similarity score. However, many small clusters in C are usually noises. This leads to incorrect similarity results. Second, with the current concept clustering algorithm, some terms can have both a general sense and a more specific sense. For example, the term "lunch" has a specific sense called "dish" and a more general (and also vague) sense called "activity". We know "activity" is more general because it is a super-concept of "dish" in Γ_{isA}. Such general senses poses problems because they make almost unrelated terms similar. For example, the term "music" also has the "activity" sense and thus is deemed similar to "lunch".

To overcome these problems, we adopt an optimization technique called *cluster pruning* after concept clustering. First, to reduce the negative impact from noisy clusters, we prune away those clusters with only one member or with very small combined weight. The weights of clusters are computed below. Let the concept clusters of the term t be $C^t = \{C_1^t, ..., C_m^t\}$, the weight of each cluster C_x^t is

$w_x / \sum_x w_x$, where $w_x = \sum_{c_i \in C_x^t} p(c_i|t)$ and $1 \leq x \leq m$. Second, to avoid the impact from the vague senses, we prune the clusters whose senses are super-concepts of other senses according to the isA relationships in Γ_{isA}. For example, Figure 4 shows a hierarchical isA relationships of senses after clustering concept contexts of two terms "lunch" and "music". Because the senses "Activity", "Cost", "Interest" and "Art" are the super-concepts of the senses "Dish" and "Multimedia", we only keep specific senses like "Dish" and "Multimedia" and remove the rest.

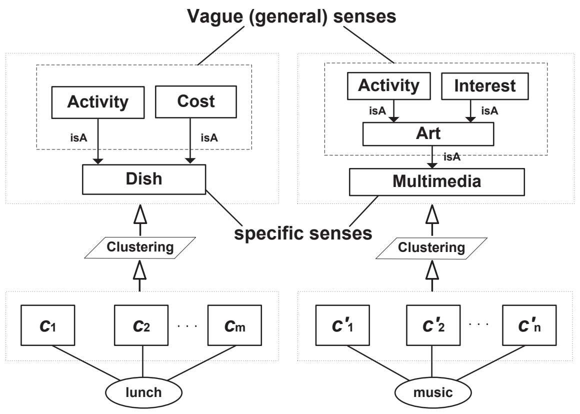

Figure 4: Illustration to vague and specific senses of terms lunch and music

5. EXPERIMENTS

In this section, we first outline the experimental setup, and then compare the effectiveness of the online and the offline variant of our approach, and also compare our approaches (basic, refined and refined with pruning) with 12 competing methods on three benchmark data sets. Finally, we evaluate the efficiency of our approaches.

5.1 Experiment Setup

We use three data sets in the following experiments, including two well-known benchmark data sets for word similarity and one labeled data set for evaluating MWEs which is created by us. Table 2 shows the descriptions and some examples in each of the three data sets. M&C data set is a subset of Rubenstein-Goodenough's [23] and consists of 28 word pairs. WordSim203 is a subset from WordSim353[1], and has been used as a similarity testing data set by Agirre et. al.[7] It contains 203 pairs which are considered more similar than related. Because there are no benchmark data for the semantic similarity between MWEs, we labeled 300 pairs (known as WP) with both words and MWEs. Our labeled data consist of three categories: 100 concept-entity pairs, 100 concept-concept pairs and 100 entity-entity pairs. These 300 pairs contain 84 word pairs and 216 MWE pairs, in which 71 MWE pairs are in WordNet the remaining are not. Five native speakers of English labeled these pairs according to the label classes, and the labels are then translated into numerical similarity scores in Table 3. These scores are averaged to produce the final rating for each pair.

All experiments are performed on an Intel Core 2 Duo 2.66GHz PC with 4G physical memory, running Windows 7 Enterprise. All timing results are averaged over 10 runs. All competing methods involved in this section are summarized in Table 4. We implemented Sán method and while adopting the existing implementation [2] of other methods. To evaluate the effectiveness of each method, we compute the Pearson Correlation Coefficient (PCC in short) to measure the agreement between the machine rating (computed by the semantic similarity measurement approaches) and the human ratings over the data sets as follows, where X is the machine ratings while Y is the human ratings:

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

Table 2: Data Sets Used in Experiments

M&C data set[17]	WordSim203 similarity [7](WS in short)	Our labeled data (WP in short)
Type		
Words	Words	Words & MWEs
#Pairs		
28	203	300
Examples		
$\langle lobster, food \rangle$	$\langle lobster, wine \rangle$	$\langle animal, poodle \rangle$
$\langle chord, smile \rangle$	$\langle professor, doctor \rangle$	$\langle microsoft, apple \rangle$
$\langle bird, cock \rangle$	$\langle tiger, jaguar \rangle$	$\langle$*shell, exxon mobil corp.*$\rangle$
$\langle$*crane, implement*$\rangle$	$\langle$*precedent,information*$\rangle$	$\langle$*caged animal, game animal*$\rangle$

Table 3: Label Classes and Similarity Scores

Label Classes	Similarity Score
Very similar	1
Fairly similar	0.75
Don't know	0.5
Fairly different	0.25
Very different	0

Table 4: Competing Methods (IC = Information Content, LCA = Least Common Ancestor)

Approach	Description
Hungarian (Hun) [3]	string-based
Tray (Tra)[4]	string-based+WordNet
Rada (Rad) [20]	path-based (WordNet)
Hirst (Hir) [14]	lexical chain-based (WordNet)
Do [13]	lexical chain-based (WordNet)
Resnik (Res)[22]	IC of LCA +WordNet
Jcn [15]	IC of LCA + the term + WordNet
Lin [16]	IC of LCA + the term + WordNet
Sánchez (Sán)[24]	IC of leaves and parents + WordNet
Banerjee (Ban)[9]	glosses-based (WordNet)
Agirre (Agi)[5]	personalized PageRank (WordNet)
Bollegala (Bol)[10]	search-snippet-based
Basic	Our basic approach
RC	Our refined approach
RCP	Our refined approach with pruning

5.2 Effectiveness

Figure 5 reports the PCC in our RCP approach with online clustering and offline clustering respectively on the WP data set. From this figure, we can see that the PCC values for online clustering and offline clustering differ only

Table 5: Pearson Correlation Coefficient on Three Data Sets with Word or MWE Pairs (WN: WordNet)

	Word Pairs				MWE Pairs		
Method	M&C	WS	Words from WP	All Word Pairs	MWEs in WN	MWEs Not in WN	All MWE Pairs
Hun	-0.20	0.06	0.02	0.04	0.37	0.43	0.36
Tra	0.76	0.59	0.38	0.48	0.39	0.33	0.34
Rad	0.74	0.60	0.40	0.51	0.59	-	-
Hir	0.64	0.57	0.45	0.51	0.46	-	-
Do	0.68	0.48	0.32	0.42	0.36	-	-
Res	0.76	0.67	0.42	0.57	0.74	-	-
Jcn	0.85	0.37	0.38	0.28	0.38	-	-
Lin	0.82	0.67	0.45	0.58	0.72	-	-
Sán	0.87	0.69	0.64	0.66	0.74	-	-
Ban	0.78	0.65	0.43	0.56	0.38	-	-
Agi	0.80	0.58	0.26	0.34	0.38	-	-
Bol	0.83	0.56	0.48	0.52	0.59	0.51	0.50
Basic	0.78	0.58	0.39	0.43	0.31	0.45	0.44
RC	0.89	0.69	0.46	0.49	0.65	0.64	0.60
RCP	**0.92**	**0.73**	**0.81**	**0.77**	**0.82**	**0.67**	**0.67**

marginally. Therefore, in the following experiments, we use the offline clustering in our refined approach.

Table 5 compares the PCC of our approaches with that of 12 others. Some of these competing methods (from Rad to Agi) rely on WordNet and do not recognize MWEs that are not in WordNet, therefore they are excluded from comparison in the experiment on "MWEs Not in WordNet", marked with "-". From the experimental results, we make the following observations.

First, our most advanced approach, RCP, leads the competition against the peers by significant margins in all data sets, especially in MWE pairs.

Second, in the Hun method, the PCC value is negative, because it depends only on the surface forms of terms. Most terms which are semantically similar are not lexically similar. Thus, some of the computed similarities are incorrect , which leads to the negative correlation.

Third, methods based on taxonomy structure, such as Rad, Hir and Do, generally fare better than pure syntax-based methods.

Fourth, information content based methods, such as Res, Jcn, Lin and Sán, generally do better than other WordNet based methods. Information content based methods effectively combines the knowledge from the taxonomy structure and external corpora. This has certain advantage but the coverage of this knowledge is still limited compared to the knowledge we acquired from the entire web.

Finally, search snippet-based method like Bol works fine with M&C data sets but fares quite badly elsewhere. This is because it considers co-occurrences of two terms which produces more of relatedness than similarity. It works badly with words in WP because word pairs in WP contain many ambiguous terms, and many pairs with transitive isA relationships (e.g., $\langle animal, puppy\rangle$ with "dog" being the child of "animal" and parent of "puppy") and many pairs with vague senses (e.g., $\langle music, lunch\rangle$ with the vague sense "activity"). Co-occurrence alone is not effective on these pairs.

Figure 6 reports the PCC of six approaches which work with arbitrary MWEs and the experiment is done on all 300 pairs from the WP data set. RCP produces a PCC value of around 0.7 which is much higher than the other peers.

Figure 7 reports the PCC of our approaches on three types of pairs in WP. From the experimental results, we can see that our three approaches have the same PCC value (0.74) on the concept pairs, because they have the same calculation mechanism on these pairs. Our methods generally work better with concept-entity pairs than entity-entity pairs. The reason is that concept-entity pairs are similar only if they are in a hypernym-hyponym relation so the similarity is clearly defined. In the case of entity-entity pairs, comparing their concept contexts can be difficult due to i) the ambiguity in the senses and ii) the noises in the super-concepts which can be very abstract and vague.

Table 6 shows some examples from each data set along with the computed similarity scores by the RCP approach. Human ratings have been uniformly normalized to [0, 1] in this table. Complete set of results can be found at `http://adapt.seiee.sjtu.edu.cn/similarity/`.

Table 6: Example Pairs and Their Semantic Similarity Scores Computed by RCP Approach

Pair	Human Rating	Similarity Score
From M&C Data Set		
$\langle furnace, stove\rangle$	0.778	0.950
$\langle bird, cock\rangle$	0.763	0.824
$\langle boy, lad\rangle$	0.940	0.800
$\langle coast, shore\rangle$	0.925	0.800
$\langle bird, crane\rangle$	0.743	0.564
$\langle lobster, food\rangle$	0.223	0.525
$\langle crane, implement\rangle$	0.420	0.294
$\langle monk, oracle\rangle$	0.275	0.002
$\langle journey, car\rangle$	0.290	0.001
$\langle chord, smile\rangle$	0.033	0.000
From WS Data Set		
$\langle tiger, jaguar\rangle$	0.800	0.979
$\langle professor, doctor\rangle$	0.662	0.930
$\langle vodka, brandy\rangle$	0.813	0.929
$\langle journey, voyage\rangle$	0.929	0.800
$\langle travel, activity\rangle$	0.500	0.532
$\langle consumer, energy\rangle$	0.475	0.518
$\langle man, governor\rangle$	0.525	0.506
$\langle reason, hypertension\rangle$	0.231	0.036
$\langle precedent, information\rangle$	0.385	0.011
$\langle lobster, wine\rangle$	0.570	0.000
From WP Data Set		
$\langle caged\ animal, game\ animal\rangle$	0.850	0.996
$\langle business, restaurant\rangle$	0.550	0.938
$\langle shell, exxon\ mobil\ corp.\rangle$	0.850	0.814
$\langle animal, poodle\rangle$	0.800	0.720
$\langle date, asian\ pear\rangle$	0.500	0.711
$\langle range, food\ processor\rangle$	0.750	0.689
$\langle climacteric\ fruit, vegetable\ juice\rangle$	0.600	0.226
$\langle music, lunch\rangle$	0.100	0.012
$\langle banana, beef\rangle$	0.350	0.007
$\langle apple, ipad\rangle$	0.200	0.006

5.3 Efficiency

Figure 8 compares the execution time between online clustering and offline clustering in our refined approach. Offline clustering, with only a fraction of the cost, is a clear winner.

Figure 9 reports the average computation time on a pair of terms in our approaches compared to the other competitors. On average, RCP takes 65 milliseconds to compute the similarity of a pair, which is on par with most of the earlier methods using information content and WordNet. String-based methods are faster for an obvious reason: they need not collect any context or model the context. Hir is slow because it considers the lexical chain in the taxonomy in the calculation of semantic similarity between terms. Bol

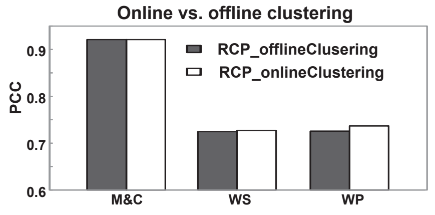

Figure 5: Performance of RCP with online/offline clustering

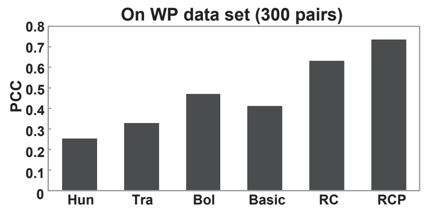

Figure 6: Performance comparison on WP

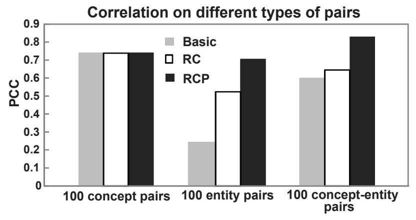

Figure 7: Performance comparison on various types of pairs

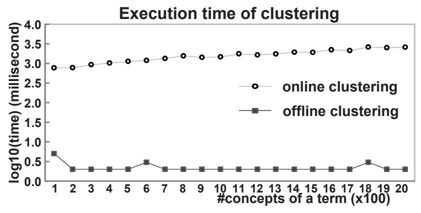

Figure 8: Execution time in online/offline clustering

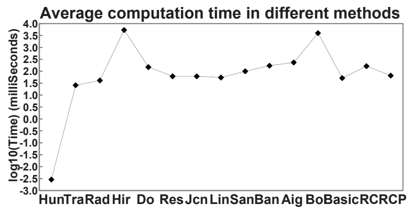

Figure 9: Computation time in different approaches

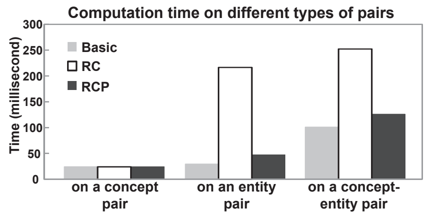

Figure 10: Computation time on different types of pairs

takes about 60 times longer than RCP because it requires extracting lexico-syntactic patterns from snippets online.

Figure 10 shows the average computation time on different types of pairs using our approaches. RCP costs less than half the time of RC because due to the pruning. Computing similarity between concept-entity pairs is more expensive because in order to catch the concept-entity pairs with potentially transitive isA relationships, e.g., "animal" and "puppy" (with "dog" being the child of "animal" and parent of "puppy"), we iteratively check the relations between every top ancestor concepts of an entity term and the concept term in RCP.

6. RELATED WORK

Contrary to the semantic relatedness which represents the more general relationships such as part-whole and the co-occurrence, semantic similarity measures the degree of taxonomic likeness between concepts and considers relations such as hyperonymy and synonymy. In this section, we only discuss previous work on semantic similarity, while most of them can be adapted or generalized to deal with semantic relatedness. To compute the semantic similarity between terms, existing efforts mainly follow two approaches: The first approach calculates the semantic similarity based on some distance in a preexisting thesauri, taxonomy or encyclopedia, such as WordNet. The second approach computes similarity by the terms' context in large text corpora (such as the search snippets and web documents) and such similarities are derived from distributional properties of words or n-grams in the corpora.

6.1 Knowledge-based Approach

Most methods in this direction use a taxonomy such as WordNet, which is a tree hierarchy, as the knowledge base to compute the similarity between terms. The most straightforward way to calculating similarity between two terms on the WordNet is to find the length of the shortest path connecting the two terms in the taxonomy graph[20]. This path-length based approach is very simple, but has a low accuracy because: i) it relies on the notion that all links in the taxonomy represent a uniform distance; ii) it ignores the amount of information hidden in the concept nodes.

More advanced approaches [22, 15, 16, 25, 24] compute the similarity between t_1 and t_2 by the information content of these terms with respect to the taxonomy structure. The pioneer work by Resnik [22] suggests that the similarity measure is the information content of the least common ancestor node of the two terms in the taxonomy tree. To compute the information content of a term, it requires a large text corpus to obtain the occurrences of the term. A limitation of this method is that the similarities between all children of a concept are identical, regardless of their individual information content. The most recent information content based approach [24] calculates the information content of term t by the ratio of the number of hypernyms of t divided by the number of all descendants of t in WordNet.

Other researchers attempted to apply graph learning algorithms on term similarity computation. Given two terms t_1 and t_2, Alvarez and Lim [8] build a rooted weighted graph called $Gsim$, using the terms hypernyms, other relations, and descriptive glosses from WordNet, and then calculate the similarity score by selecting the minimal distance between any two hypernyms c_1 and c_2 of t_1 and t_2 respectively, by random walk. Agirre et. al. subsequently proposed a WordNet-based personalized PageRank algorithm [6, 5]. It first computes the personalized PageRank of each word and aggregates into a probability distribution for each synset. Similarity is then defined by the cosine between two distributions.

The above knowledge based approaches depend heavily on the completeness of the underlying taxonomy and the external corpora. However, the popular taxonomy like WordNet does not have the adequate coverage as it cannot keep up with the development of new terms and phrases everyday.

The framework proposed in this paper is also knowledge based, but is more scalable and effective, because i) the knowledge we use was acquired from the entire Web; and ii) the clustering algorithm detects the senses of the input terms and the max-max similarity function effectively picks the senses that are most suitable given the pair of terms. The above methods cannot be easily adapted to use Probase because it is a general network, not a tree structure.

6.2 Corpus-based Approach

In this space, Chen et. al. proposed a double-checking model using text snippets returned by a Web search engine to compute semantic similarity between words [12]. The proposed method uses the occurrences of terms X and Y in their search snippets to evaluate the semantic similarity. Recently, Bollegala et.al. proposed a new measure using page counts and snippets from Web search [10]. The search engine based methods are more time-consuming because i) snippets and search results must be obtained online; ii) it requires parsing of the returned text by the patterns.

Radinsky et. al. proposed a new model, Temporal Semantic Analysis (TSA) [21], which captures the temporal information of corpus. TSA uses a more refined representation, where each concept is no longer scalar, but is instead represented as time series over a corpus of temporally-ordered documents. This method can improve the pearson correlation coefficient, but it requires massive historical data.

Most corpus based methods are more suitable for the semantic relatedness not for the semantic similarity because they make heavy use of the co-occurrence context in the representation of terms or in similarity functions.

7. CONCLUSIONS

We presented a lightweight, effective approach for semantic similarity between terms with any multi-word expression. Our clustering-based refined algorithm outperforms the state-of-the-art methods as well as our basic algorithm in terms of pearson correlation coefficient on word pairs and MWE pairs. The method is efficient enough to be applied on large scale data sets.

8. REFERENCES

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

[2] `http://wn-similarity.sourceforge.net/`.

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

[4] `http://www.codeproject.com/Articles/11835/Word-Net-based-semantic-similarity-measurement`.

[5] E. Agirre, M. Cuadros, G. Rigau, and A. Soroa. Exploring knowledge bases for similarity. In *Proceedings of LREC'10*, pages 373–377, 2010.

[6] E. Agirre and A. Soroa. Personalizing pagerank for word sense disambiguation. In *Proceedings of EACL'09*, pages 33–41, 2009.

[7] E. Agirre, A. Soroa, E. Alfonseca, K. Hall, J. Kravalova, and M. Pasca. A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proceedings of NAACL'09*, pages 19–27, 2009.

[8] M. Alvarez and S. Lim. A graph modeling of semantic similarity between words. In *Proceedings of the Conference on Semantic Computing*, pages 355–362, 2007.

[9] S. Banerjee and T. Pedersen. An adapted lesk algorithm for word sense disambiguation using wordnet. In *Proceedings of CICLING'02*, pages 136–145, 2002.

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

[11] A. Budanitsky and G. Hirst. Evaluating wordnet-based measures of lexical semantic relatedness. *Computational Linguistics*, 32:13–47, 2006.

[12] 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.

[13] Q. Do, D. Roth, M. Sammons, Y. Tu, and V. Vydiswaran. Robust, light-weight approaches to compute lexical similarity. Technical report, 2009.

[14] G. Hirst and D. St-Onge. Lexical chains as representations of context for the detection and correction of malapropisms. In *WordNet: An Electronic Lexical Database*, pages 305–332, 1998.

[15] J. Jiang and D. Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of International Conference on Research in Computational Linguistics*, pages 19–33, 1997.

[16] D. Lin. An information-theoretic definition of similarity. In *Proceedings of ICML'98*, pages 296–304, 1998.

[17] G. Miller and W. Charles. Contextual correlates of semantic similarity. *Language and Cognitive Processes*, 6:1–28, 1998.

[18] G. A. Miller. WordNet: A lexical database for english. *Commun. ACM*, 38(11):39–41, 1995.

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

[20] R. Rada, H. Mili, E. Bichnell, and M. Blettner. Development and application of a metric on semanticnets. *IEEE Transactions on Systems, Man and Cybernetics*, 9:17–30, 1989.

[21] K. Radinsky, E. Agichtein, E. Gabrilovich, and S. Markovitch. A word at a time: Computing word relatedness using temporal semantic analysis. In *Proceedings of WWW'11*, pages 337–346, 2011.

[22] P. Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of IJCAI'95*, pages 448–453, 1995.

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

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

[25] N. Seco, T. Veale, and J. Hayes. An intrinsic information content metric for semantic similarity in wordnet. In *Proceedings of ECAI'04*, pages 1089–1090, 2004.

[26] W. Wu, H. Li, H. Wang, and K. Q. Zhu. Probase: a probabilistic taxonomy for text understanding. In *Proceedings of SIGMOD'12*, pages 481–492, 2012.