A Large Probabilistic Semantic Network Based Approach to Compute Term Similarity

Peipei Li, Haixun Wang, Kenny Q. Zhu, Zhongyuan Wang, Xuegang Hu, and Xindong Wu, *Fellow, IEEE*

Abstract—Measuring semantic similarity between two terms is essential for a variety of text analytics and understanding applications. Currently, there are two main approaches for this task, namely the knowledge based and the corpus based approaches. 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. Contrary to these existing techniques, we propose an efficient and effective approach for semantic similarity using a large scale semantic network. This semantic network is automatically acquired from billions of web documents. It consists of millions of concepts, which explicitly model the context of semantic relationships. In this paper, we first show how to map two terms into the concept space, and compare their similarity there. Then, we introduce a clustering approach to orthogonalize the concept space in order to improve the accuracy of the similarity measure. Finally, we conduct extensive studies to demonstrate that our approach can accurately compute the semantic similarity between terms of MWEs and with ambiguity, and significantly outperforms 12 competing methods under Pearson Correlation Coefficient. Meanwhile, our approach is much more efficient than all competing algorithms, and can be used to compute semantic similarity in a large scale.

Index Terms—Term similarity, multi-word expression, clustering, semantic network

1 Introduction

Computing semantic similarity between terms is a fundamental problem in lexical semantics [1] and it finds many applications in web and document search [2], [3], 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 [4] as shown in Fig. 1. Semantic similarity is a more specific relationship and is much harder to model than *relatedness* (which can be modeled by term co-occurrence).

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 [5], [6], [7] depends on the semantic isA relations in WordNet 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 webpages, web search snippets to other text repositories.

However, one significant challenge faced by the knowledge-based methods is the limited coverage of taxonomies such as WordNet. Through two decades of development, the most recent release of WordNet (version 3.0) contains 155,287 words organized in 117,659 synsets and 206,941 word-sense pairs. Still, 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". Thus, it is impossible for these WordNet-based methods to correctly compute the semantic similarity which involves these unknown terms or terms with unknown senses. For example, the similarity between "General Electric" and "GE" completely fail even though they are exactly the same thing, that is, "GE" has one meaning as the abbreviation of "General Electric". With today's

- *P. Li is with the School of Computer Science and Information Engineering, Hefei University of Technology, Hefei 230009, China. E-mail: peipeili@hfut.edu.cn.*
- *H. Wang is with Google Research, Mountain View, CA 94043 USA. E-mail: haixun@google.com.*
- *K.Q. Zhu is with the Department of Computer Science, Shanghai Jiao Tong University, Shanghai 200240, China. E-mail: kzhu@cs.sjtu.edu.cn.*
- *Z. Wang is with Renmin University of China and Microsoft Research Asia, Beijing, China. E-mail: zhy.wang@microsoft.com.*
- *X. Hu is with the School of Computer Science and Information Engineering, Hefei University of Technology, Hefei, Anhui Province, China. E-mail: jsjxhuxg@hfut.edu.cn.*
- *X. Wu is with the School of Computer Science and Information Engineering, Hefei University of Technology, China, and the Department of Computer Science, University of Vermont, Burlington, VT 05405. E-mail: xwu@uvm.edu.*

Manuscript received 3 Apr. 2014; revised 18 Mar. 2015; accepted 19 Mar. 2015. Date of publication 2 Apr. 2015; date of current version 8 Sept. 2015. Recommended for acceptance by H. Xiong.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TKDE.2015.2419673

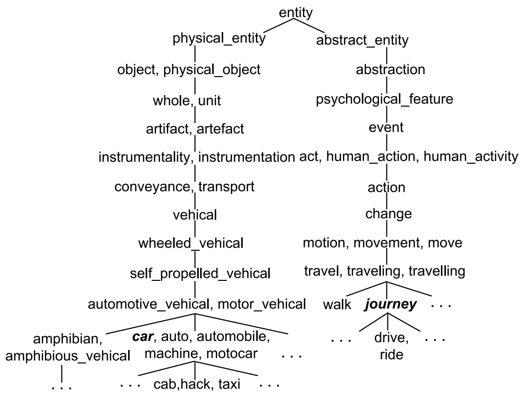

Fig. 1. A fragment from WordNet showing semantic distance between "car" and "journey".

fast changing world, it is just not possible for manually curated lexical databases like WordNet to keep up with the pace of the creation of new words and phrases in human languages.

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-based 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. e.g., "apple" can be both a fruit and a company. Consequently, distribution thus computed may not truly represent the semantic landscape of the contexts. 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 an efficient and effective framework for computing semantic similarity (a number between 0 and 1) between two 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*$\rangle$

  Synonyms that refer to the same entity should have the highest similarity score.
- High similarity (ambiguous terms): $\langle$*microsoft*, *apple*$\rangle$, $\langle$*orange*, *red*$\rangle$

  Words such as "apple" and "orange" have multiple senses. However, when people compare "apple" with "microsoft", they consider "apple" in the sense of a company rather than a fruit, and when they compare "orange" and "red", they consider "orange" as a color rather than a fruit. Thus, disambiguation needs to be performed by default in similarity comparison.
- Low similarity (though share same hypernyms in WordNet): $\langle$*music*, *lunch*$\rangle$, $\langle$*banana*, *beef*$\rangle$

  These pairs of terms are not similar. However, in an isA network, "music" and "lunch" may both belong to concepts such as "activity", and "banana" and "beef" may both belong to concepts such as "food". We may use their distances in a handcrafted taxonomy to measure similarity, but handcrafted taxonomies have low coverage, while distances in large scale, data driven semantic networks are not easy to measure.
- Low similarity (related but not similar): $\langle$*apple*, *ipad*$\rangle$, $\langle$*car*, *journey*$\rangle$

  We need to differentiate similarity from relatedness. Here, "apple" and "ipad", "car" and "journey" are related, but they are not similar. This is because "ipad" is an electronic product of the company "apple" while "car" is a traffic tool for the activity "journey", however, they belong to the different concepts or far away on an isA taxonomy.

Our main contributions of this paper are below.

- *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. Unlike existing methods based on WordNet which only measure the similarity between limited number of words, 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 more efficient.* The most expensive clustering algorithm is performed offline. The remaining similarity function can be efficiently computed online. On average, it takes 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. Section 5 gives the experimental studies. 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 [8]. Besides other knowledge, Probase contains isA relations between concepts, subconcepts, and entities, called Γ_{isA} in this paper. The isA relationships in Probase are harvested from 1.68 billion webpages and two years' worth of Microsoft Bing's search log using syntactic patterns (e.g., the Hearst patterns [9] and the is-a pattern, both are collectively named as isA extractions). Information in Γ_{isA} is in the form of (c, e, W), where c is a hypernym and e is a hyponym, $\langle c, e\rangle$ is a pair of hypernym and hyponym, for instance, $\langle country, usa\rangle$, and W is a set of probabilistic scores. Two of the most important scores in W are known as *typicality*: $P(e|c)$, the typicality of e of category c, and $P(c|e)$, the typicality of category c for e. Both scores are approximated by frequencies, e.g.,

$$P_{\Gamma_{isA}}(e|c) = \frac{N_{\langle c,e\rangle} \text{ in isA extraction}}{N_c \text{ in isA extraction}},$$

where $N(\cdot)$ indicates the occurrences of the given terms or term pairs. 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. In sum, Probase has the following 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. For example, the term "banana" is connected to concepts such as "fruit" and "tree" directly. The benefit is that such links are data driven rather than handcrafted. There is no need to transitively find all super concepts, but the distance between two terms cannot be easily measured by the number of steps it takes to reach each other.
- 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" versus "2d bar code";
- *singular/plural forms*: "shoe" versus "shoes";

We address this issue in two steps. First, we use the available sources such as Wikipedia Redirects and 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 below.

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

If $d_{lex}(t_1, t_2) < \varphi$, the two terms in the current pair are ones with very similar surface forms, and we group them together. The edit distance based method is very simple while the performance is closely relevant to the value of φ, namely the smaller of the value of φ the better of the performance. In this paper, we set φ to 0.05 according to empirics, which enables high accuracy (95 percent) for identifying synonymous pairs in our knowledge base.

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 mapped logically into isA relations between synsets. The set of all synsets called Γ_{ssyn} 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, i.e., 1.

3 Basic Approach

This section presents the basic framework of computing semantic similarity between two terms. In a 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

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

where $sim(\cdot)$ is a similarity function for contexts.

3.1 Type Checking

A basic step in the measuring of semantic similarity between terms is to decide the types of given terms, namely checking the given term is an entity or a concept. 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: it is a concept if its frequency as a hypernym in Γ_{isA} is larger than its frequency as a hyponym; it is an entity otherwise. The above method of type checking is implemented when no text context of the given pair of terms is available. If the context is available, we can introduce the $Conceptualization$ method [10] to determine which sense the given pair belongs to. For example, from the context of "Apple, Microsoft and Google are World's most valuable brands", we may derive the concept of company given the pair of "apple, microsoft". However, this is beyond the scope of this paper. Here we only consider the similarity between two terms of a pair without textual contexts.

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. Thus, we assume we have the following data: i) For any entity term e, we are given the set of concepts that e belongs to. For example, Microsoft may belong to the concepts such as company, client, large company and industry leader. ii) For any concept term c, we are given the set of entities that c subsumes. For example, Country may contain entities such as china, germany, australia, japan, france and usa. iii) For any pair of entity e and concept c, we know how typical e is as an entity for c and how typical c is as a concept for e. For instance, people may think of Arnold Schwarzenegger as a movie star, a politician, a bodybuilder, a businessman, or an investor. But the weight (typicality) of Arnold being a movie star is higher than being an investor.

From the above information, we could derive the following context vectors for entity term e and concept term c:

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

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.

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

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 similarity function $F(\cdot)$ to evaluate the similarity between two contexts, i.e.,

$$sim(T_{t_1}, T_{t_2}) = F(T_{t_1}, T_{t_2}). \tag{4}$$

$F(\cdot)$ can be one of the popular similarity evaluation functions, such as cosine and Jaccard. Algorithm 1 shows the complete algorithm for the basic approach.

Algorithm 1. Basic 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 $\Gamma_{isA}$;
       $maxD$: the maximum iteration depth;
Output: a similarity score of ⟨$t_1, t_2$⟩;
 1: if $t_1$ and $t_2$ belong to the same synset by $\Gamma_{ssyn}$ 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 ⟨$t_1, t_2$⟩ is a concept pair then
 6:     Generate the entity vector $\mathcal{I}_c^{t_i} (i \in \{1, 2\})$ of $t_i$ as defined
        in (2) using $\Gamma_{isA}$;
 7:     return $sim(\mathcal{I}_c^{t_1}, \mathcal{I}_c^{t_2})$ as defined in (4);
 8: end if
 9: if ⟨$t_1, t_2$⟩ is an entity pair then
10:     Generate the concept vector $\mathcal{I}_e^{t_i} (i \in \{1, 2\})$ of $t_i$ as defined
        in (3) using $\Gamma_{isA}$;
11:     return $sim(\mathcal{I}_e^{t_1}, \mathcal{I}_e^{t_2})$ as defined in (4);
12: end if
13: if ⟨$t_1, t_2$⟩ is a concept-entity pair then
14:     Collect topK concepts of the entity term $t_i$ from $\Gamma_{isA}$ as the
        context $C_{t_i} (i \in \{1, 2\})$;
15:     for each $c_x$ in $C_{t_i}$ ($c_x \neq t_j, i \neq j, 1 \leq x \leq topK$) do
16:         $sim_{c_x} \leftarrow$ get the similarity between $c_x$ and $t_j$ by repeating this algorithm iteratively if the current iteration
            depth is no more than $maxD$;
17:     end for
18:     return $\max_{c_x \in C_{t_i}} \{sim_{c_x}\}$;
19: end if
```

TABLE 1
Impact of Ambiguity on Similarity (S.S. = Similarity Score)

Pair	S.S.	Term	Main Sense	Prob.
		microsoft	company	0.825
			search engine	0.525
		google	company	0.342
			fruit	0.441
⟨*microsoft*, *google*⟩	0.993		company	0.235
⟨***apple***, *pear*⟩	0.916	*apple*	food	0.104
⟨***apple***, *microsoft*⟩	**0.378**		tree	0.068
⟨***orange***, *red*⟩	**0.491**		fruit	0.856
		pear	tree	0.120
			fruit	0.456
		orange	color	0.293
			food	0.078
		red	color	0.926

3.4 Discussion

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. For example, as shown in Table 1, the basic approach decides that ⟨*microsoft*, *google*⟩ and ⟨*apple*, *pear*⟩ are quite similar whereas ⟨*apple*, *microsoft*⟩ and ⟨*orange*, *red*⟩ are not, because "apple" and "orange" have multiple senses. Table 1 lists main senses of the given terms whose probabilities are higher than 0.05. We can see that the dominant senses of "apple" and "orange" are a fruit. When we are comparing similarity using non-dominant senses, the results are hence less satisfactory.

4 Refined Approach

The basic approach 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 refined 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. We select k-Medoids for the following reasons. First, the k-series clustering algorithms are simple and effective. Second, contrary to the algorithms of k-Means, k-Medians, or k-Modes with virtual centroids, k-Medoids can get the accurate centroids, that is, each cluster has an existing centroid (concept) as the sense of the current cluster.

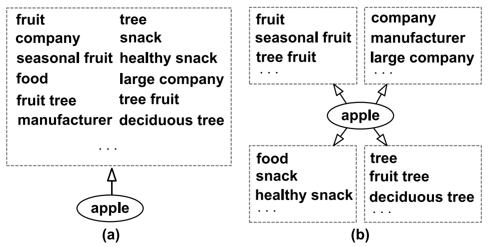

Fig. 2. The concept context of “apple”.

Fig. 2a shows the concept context of the term “apple” in Probase, and Fig. 2b shows the clustered concepts. It is clear that each cluster represents a sense of the term.

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 (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 [11]. 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.,

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

where c_j is the jth candidate point, m_i is the ith medoid in existing medoids, and α is the threshold in the limit of initial medoid count. This process continues until we do not find any medoids satisfying (6). In this case, we get k medoids at iteration 0: $M^0 = \{m_1^0, \ldots, m_k^0\}$. Clearly, the larger the threshold of α, the small the value of k.

With k medoids in the tth iteration, we assign each candidate concept $c_i \in C$ to its closest medoid $m^* \in M^t = \{m_1^t, \ldots, m_k^t\}$, 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, we first compute the average distance of a cluster K_i in terms of the semantic distance in (5) as

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

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 ith cluster medoid.

We use (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}). According to the above processing of k-Medoids, we can get k clusters for all given concepts. The minimization of F with the above constraints is an undecidable constrained nonlinear optimization problem. A partial optimization for M and W is shown in Algorithm 2.

Algorithm 2. Concept Clustering

```
Input: $C = \{c_1, \ldots, c_j, \ldots\}$: the concept set;
       $\alpha$: the threshold relevant to initial medoid count;
       $T$: the maximum iteration count;
       $\Gamma_{isA}$: the semantic network of isA relationship;
Output: $k$ clusters $\{K_1, \ldots, K_k\}$;
 1: Initialize the iteration time $t = 0$;
 2: Generate an initial medoid set $M^t = [m_1^t, m_2^t, \ldots, m_k^t]$ incrementally in (6);
 3: Assign each concept $c_i$ to a cluster $K^*$ with a medoid $m^*$ satisfying (7);
 4: Update the weight matrix $W^t$ in (10) to make sure $F(W^t, M^t)$ is minimum;
 5: Update cluster medoids in $M^{t+1}$ using (8);
 6: Calculate $F(W^t, M^{t+1})$ in (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 $\{K_1, \ldots, K_k\}$;
```

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 $O(n)$.

To cluster the concepts in the semantic network, we first cluster the topK popular concepts (e.g., topK = 180,000) into k clusters. We then allocate the remaining concepts using Eq. (5) into the corresponding clusters. Finally, we

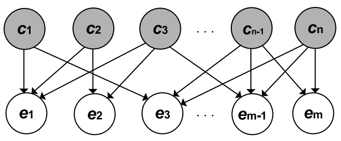

Fig. 3. A concept-entity bipartite graph.

iteratively repeat the above processing for each cluster until the constraint is met (e.g., the total number of clusters). In the clustering, we use the entity distributions to represent the concepts and evaluate their similarities in (5). According to the isA relationships between concepts and entities in Γ_{isA}, we can construct a bipartite graph between concepts and entities (Fig. 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 a L2-normalized vector $\mathcal{I}_{c_i}$ as shown in (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 Methods of Similarity Computation

In the basic approach, we compute the similarity of two terms by a similarity function between their textual contexts. In the refined approach, we explore three new methods for similarity computation, known as *max*, *average*, *weighted similarity*. Let $K = \{K_1, \ldots, K_k\}$ 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. Correspondingly, we can get the clusters in C_{t_1} and C_{t_2}, namely K_{t_1} and K_{t_2} as follows:

$$\begin{aligned} K_{t_1} &= \{x \mid x = K_i \cap sup(t_1), \forall K_i \in K \wedge x \neq \emptyset\}, \\ K_{t_2} &= \{y \mid y = K_i \cap sup(t_2), \forall K_i \in K \wedge y \neq \emptyset\}, \end{aligned} \tag{11}$$

where $sup(t_i) = \{c \mid \langle c, t_i \rangle \in \Gamma_{isA}\}$. We then compute the similarity between the contexts of each cluster pair and get the semantic similarity between two terms as:

$$\begin{aligned} Max: sim(T_{t_1}, T_{t_2}) &= Max_{x \in K_{t_1}, y \in K_{t_2}}\{F(x, y)\}, \\ Average: sim(T_{t_1}, T_{t_2}) &= 1/|K_{in}| \Sigma_{x,y \in K_{in}} F(x, y), \\ Weighted: sim(T_{t_1}, T_{t_2}) &= \Sigma_{x \in K_{in}} w_x \Sigma_{y \in K_{in}} w_y F(x, y). \end{aligned} \tag{11a}$$

Where $K_{in} = K_{t_1} \bigcap K_{t_2}$, $w_x = v_x / \Sigma_{z \in K_{t_1}} v_z$, and $w_y = v_y / \Sigma_{z \in K_{t_2}} v_z$. The corresponding value vector of x (or y) indicates the set of typicality scores for t_1 (or t_2) and each concept in $sup(t_1)$ (or $sup(t_2)$), and v_x (v_y) indicates the sum of typicality scores in the corresponding vector.

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: $\langle t_1, t_2 \rangle$: a pair of terms;
       $\Gamma_{isA}$: the semantic network of isA relationship;
       $\Gamma_{ssyn}$: the synset data set in $\Gamma_{isA}$;
       $\Gamma_{cluster}$: clusters of all concepts in $\Gamma_{isA}$;
       $maxD$: the maximum iteration depth;
Output: a similarity score of $\langle t_1, t_2 \rangle$;
 1: Install the synset checking and type checking as Steps 1-4 in
    Algorithm 1;
 2: if $\langle t_1, t_2 \rangle$ 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 $\langle t_1, t_2 \rangle$ is an entity pair then
 6:     $s_1 \leftarrow sim(\mathcal{I}_e^{t_1}, \mathcal{I}_e^{t_2})$ as Steps 10-11 in Algorithm 1;
 7:     Find clusters of contexts $K_{t_1}$ and $K_{t_2}$ from $\Gamma_{cluster}$;
 8:     $s_2 \leftarrow sim(K_{t_1}, K_{t_2})$ computed in (11a);
 9:     return $max(s_1, s_2)$;
10: end if
11: if $\langle t_1, t_2 \rangle$ 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 the clusters of contexts $K_{t_i}$ from $\Gamma_{cluster}$;
14:     for each cluster $x$ in $K_{t_i}$ do
15:         Select $topK$ concepts to represent $t_i$, namely $C_x^{topK} =$
            $\{c_y | c_y \neq t_j, c_y \in x, 1 \leq y \leq topK\}$;
16:         for each concept $c_y$ in $C_x^{topK}$ do
17:             $sim_{c_y} \leftarrow$ get the similarity between $c_y$ and $t_j$ by
                repeating this algorithm iteratively if the current
                iteration depth is no more than $maxD$;
18:         end for
19:     end for
20:     return the similarity score in (11a);
21: 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* 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* 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 pose 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 a *cluster pruning* technique 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

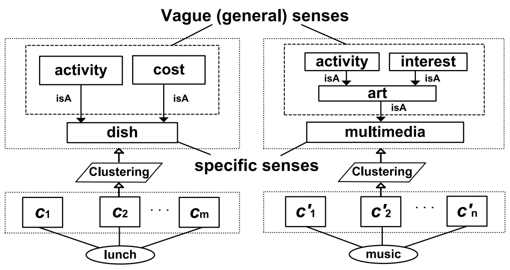

Fig. 4. Illustration to vague and specific senses of terms lunch and music.

weight. The weights of clusters are computed below. Let the concept clusters of the term t be $K^t = \{K_1, \ldots, K_m\}$, the weight of each cluster K_i is $v_i / \sum_i v_i$, where $v_i = \sum_{c_j \in K_i} p(c_j|t)$ and $1 \leq i \leq m$. 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, Fig. 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.

5 Experiments

In this section, we first give the experimental setup and some parameter analyses in Sections 5.1, 5.2, and 5.3, 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 in Section 5.4. Finally, we evaluate the efficiency of our approaches in Section 5.5.

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. M&C data set [12] is a subset of Rubenstein-Goodenough's [13]. WordSim203 as a similarity testing data set [14] is a subset from WordSim353 [15]. 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, namely "Very similar" (1), "Fairly similar" (0.75), "Don't know" (0.5), "Fairly different" (0.25), "Very different" (0). These scores are averaged to produce the final rating for each pair.

All experiments are performed on an Intel Core 2 Duo 2.66 GHz PC with 4 G main memory, running Windows XP. All timing results are averaged over 10 runs. All competing methods involved in this section are summarized in Table 2.

TABLE 2
Competing Methods
(IC = Information Content, LCA = Least Common Ancestor)

Approach	Description
Hungarian (Hun) [18]	string-based
Tray (Tra)[19]	string-based+WordNet
Rada (Rad) [5]	path-based (WordNet)
Hirst (Hir) [20]	lexical chain-based (WordNet)
Do [21]	lexical chain-based (WordNet)
Resnik (Res)[6]	IC of LCA + WordNet
Jcn [22]	IC of LCA + the term + WordNet
Lin [23]	IC of LCA + the term + WordNet
Sánchez (Sán)[24]	IC of leaves and parents + WordNet
Banerjee (Ban)[25]	glosses-based (WordNet)
Agirre (Agi)[7]	personalized PageRank (WordNet)
Bollegala (Bol)[26]	search-snippet-based
Basic	Our basic approach
RC	Our refined approach
RCP	Our refined approach with pruning

We implemented Sán method while adopting the existing implementation [16] of other methods. To evaluate the effectiveness of each method, we compute the Pearson-Correlation Coefficient (PCC in short) [17] to measure the agreement between the machine rating (computed by the semantic similarity measurement approaches) and the human ratings over the data sets.

5.2 Parameter Analysis

In this section, we describe some experiments on three important parameters (namely $topK$, $maxD$ and α) involved in our approaches and the refined k-Medoids algorithm. From previous analysis, we know that the values of $topK$ and $maxD$ are related to the computation time and the PCC value predicted on the concept-entity pairs from WP, and the value of α is related to the number of clusters and the PCC value predicted on the concept-entity pairs. Because the experimental conclusions about these three parameters are irrelevant to the concise similarity function as $F(\cdot)$, the following experiments are conducted with the cosine similarity function, namely $F(\cdot) = cosine$.

Figs. 5 and 6 report the curves of the PCC value and the mean computation time on the concept-entity pairs varying with values of $topK$ and $maxD$ from 1 to 20 respectively. Experimental results in Fig. 5 show that as the value of $topK$ increases, the mean computation time on each concept-entity pair is linearly increasing, while the PCC value first increases to a peak value (in the case of $topK \geq 3$), then decreases to a stable value. Experimental results in Fig. 6 show that as the value of $maxD$ increases, the computation time on each pair is steadily increasing up to a stable value, while the PCC value first increases to a peak value (in the case of $maxD \leq 3$), then decreases to a stable value. This is because all concept-entity pairs in our experiments have no deeply transitive isA relationships, the largest depth is no more than seven levels. According to the above analysis, to trade-off the two evaluation measures, namely maintaining a higher PCC value and lower computation time, we select the optimal value of $topK = 3$ and $maxD = 3$ respectively in the following experiments.

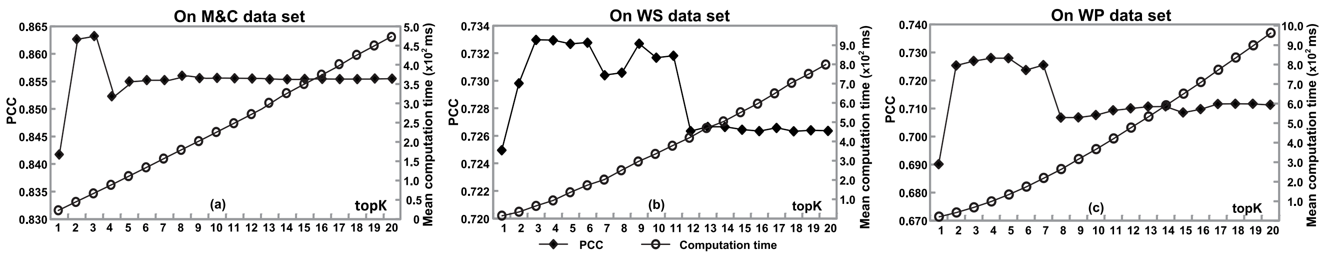

Fig. 5. PCC values and computation time varying with the values of $topK$.

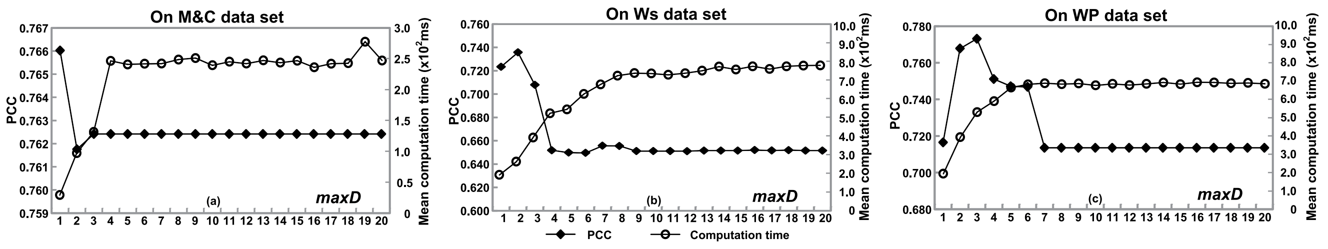

Fig. 6. PCC values and computation time varying with the values of $maxD$.

Fig. 7 reports the cluster counts and PCC values on three data sets varying with the values of α from 0.05 to 0.95 with a 0.05 step. We observe the following from the experimental results. First, as the value of α increases, the count of clusters is linearly decreasing from 29,483 to 4,628. The reason is very obvious corresponding to the constraint in (6). Second, as the count of clusters decreases, PCC values are rapidly increasing up to the peaks, and then maintaining relative stability (in the case of α varying from 0.6 to 0.8), and finally decreasing continuously. Thus, we conclude that the optimal value of α ranges from 0.6 to 0.8. In our experiments, we select the value of α =0.7 as an optimal value in the clustering algorithm of k-Medoids.

5.3 Selection on the Similarity Function

Fig. 8 reports the performance of the RCP approach using three similarity computation methods (namely Max, $Average$ and $Weighted$ in (11a)) with the cosine similarity as $F(\cdot)$. From the experimental results, we can observe that the Max method performs the best on all data sets. This is because it is more useful in identifying rare senses of terms within small clusters compared to the $Average$ and $Weighted$ methods. In the following experiments, we select the Max method of similarity computation in our refined approaches.

Fig. 9 reports the performance of the RCP approach varying with five different similarity functions as $F(\cdot)$, including *cosine*, *Jaccard*, *JaccardExt*, JS_{sim} and KL_{sim}, where *JaccardExt* indicates the extended Jaccard similarity (a.k.a. Tanimoto Similarity), JS_{sim} and KL_{sim} indicate the Jensen-Shannon divergence based and the smoothed Kullback−Leibler divergence based similarity functions respectively. Experimental results show that on the data sets of M&C and WS, RCP using KL_{sim} performs worst, while RCP using other similarity functions perform very similarly to each other on the PCC value. The largest variance of PCC values is no more than 0.01. However, on the WS data set, RCP using *cosine* and *JaccardExt* outperforms that using other three similarity functions. The PCC value is improved by 0.06 at least. These data reveal that our RCP algorithm using the similarity functions of *cosine* and *JaccardExt* is superior to others. In the following experiments, we select the cosine similarity function as $F(\cdot)$ used in (4).

5.4 Effectiveness

In this section, we aim to observe the effectiveness of our approaches in three aspects. First, we give the performance of our RCP approach using two different clustering methods, namely online clustering and offline clustering. Second, we compare our three approaches with baseline ones on the

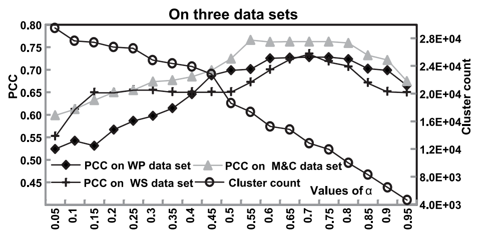

Fig. 7. Relationships between α and cluster count and PCC values.

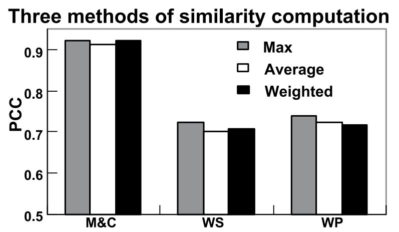

Fig. 8. Performance of three similarity computation methods.

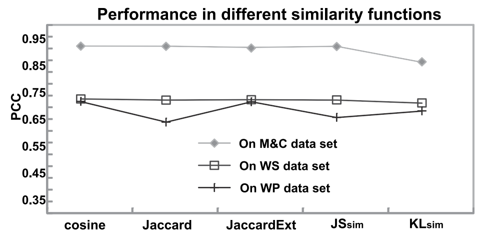

Fig. 9. Performance comparison varying with five similarity functions.

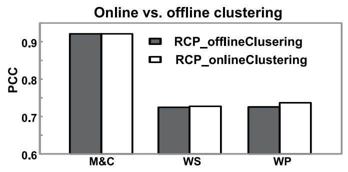

Fig. 10. Performance of RCP with online/offline clustering.

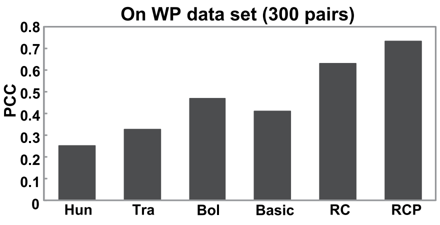

Fig. 11. Performance comparison on WP.

PCC values. Third, we further consider the performance variance among our three approaches on different pair types and on the entity disambiguity. Details are as follows.

RCP using offline clustering versus RCP using online clustering. Fig. 10 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 marginally. Therefore, in the following experiments, we use the offline clustering in our refined approach.

Performance comparison between our three approaches and baselines. Table 3 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 experiments, marked with "-". From the experimental results, we make the following observations.

First, our most advanced approach, RCP, leads the competition against the peers by large margins in all data sets, especially in MWE pairs. Second, in the Hun method, the PCC value is negative, because it only depends 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, IC based methods, such as Res, Jcn, Lin and Sán, generally do better than other WordNet based methods. IC based methods effectively combine 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 set 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., $\langle$*mammal*, *puppy*$\rangle$ with *dog* being the child of *mammal* 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.

In addition, Fig. 11 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.

Performance comparison among our three approaches. Fig. 12 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.

Meanwhile, Figs. 14 and 15 report the similarity scores produced by our three approaches against human ratings on the entity-entity pairs and concept-entity pairs from WP.

TABLE 3
Pearson Correlation Coefficient on Word Pairs, MWE Pairs, and Word + MWE Pairs

	Word Pairs				MWE Pairs			Word+MWE pairs		
Method	M&C	WS	from WP	All	in WordNet	not in WordNet	All	in WordNet	not in WordNet	All
Hun	−0.196	0.064	0.015	0.038	0.371	0.429	0.355	0.037	0.426	0.054
Tra	0.755	0.594	0.379	0.480	0.389	0.325	0.344	0.531	0.320	0.468
Rad	0.739	0.595	0.395	0.510	0.592	-	-	0.544	-	-
Hir	0.643	0.574	0.451	0.511	0.459	-	-	0.530	-	-
Do	0.676	0.482	0.322	0.419	0.359	-	-	0.423	-	-
Res	0.762	0.672	0.424	0.569	0.744	-	-	0.567	-	-
Jcn	0.848	0.371	0.382	0.275	0.382	-	-	0.107	-	-
Lin	0.822	0.674	0.446	0.579	0.717	-	-	0.452	-	-
Sán	0.865	0.690	0.643	0.655	0.740	-	-	0.585	-	-
Ban	0.781	0.651	0.426	0.560	0.377	-	-	0.545	-	-
Agi	0.795	0.579	0.258	0.343	0.380	-	-	0.344	-	-
Bol	0.834	0.564	0.476	0.523	0.592	0.511	0.498	0.521	0.509	0.505
Basic	0.777	0.576	0.387	0.429	0.313	0.449	0.440	0.508	0.508	0.494
RC	0.885	0.690	0.457	0.494	0.651	0.635	0.595	0.573	0.645	0.589
RCP	**0.921**	**0.725**	**0.811**	**0.770**	**0.822**	**0.665**	**0.670**	**0.761**	**0.683**	**0.735**

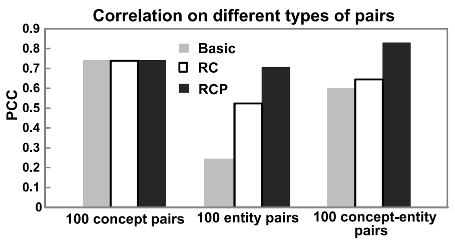

Fig. 12. Performance comparison on various types of pairs.

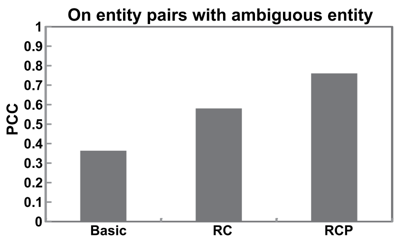

Fig. 13. Performance of entity disambiguity in our approaches.

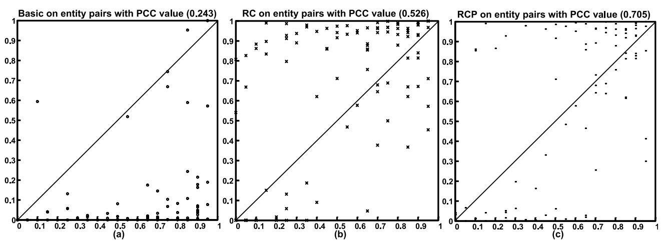

Fig. 14. Performance of our three approaches on entity pairs.

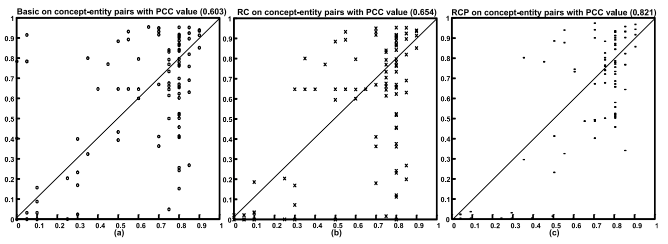

Fig. 15. Performance of our three approaches on concept-entity pairs.

We know that the more the points scatter on the $y = x$ line, the better the prediction result, namely the higher the PCC values. From the experimental results, we can see that none of the methods are linear, but RC and RCP approaches actually improve the prediction accuracy compared to Basic without the clustering method, while RCP further improves the prediction accuracy using the cluster pruning compared to RC. In addition, Table 5 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/.

Furthermore, we address the performance of RC and RCP approaches on the entity disambiguity, because disambiguating named entities is very significant and necessary [27]. To validate the performance on the entity disambiguity, our experiments are set below. First, we select 30 entity-entity pairs from WP as the test data and each entity-entity pair contains only one ambiguous entity. Second, we identify main senses of the ambiguous entities using our approaches, and then determine the dominant sense of the ambiguous entity corresponding to the sense of the other unambiguous entity, and finally get the similarity score of the pair. Third, we show the effectiveness of our RC and RCP approaches with the concept clustering in two dimensions.

TABLE 4
Prediction Results in RCP on 30 Entity Pairs with Ambiguous Entities

pairs	dominant sense in RC	dominant sense in RCP
⟨**fox**, *polar bear*⟩	**animal**	mammal
⟨**fox**, *nbc*⟩	channel /network	channel /network
⟨**apple**, *ipad*⟩	null	null
⟨*blue berry*, **apple**⟩	fruit	fruit
⟨**apple**, *microsoft*⟩	company	company
⟨**chicken**, *hen*⟩	animal	animal
⟨**chicken**, *beef*⟩	**food**	meat
⟨**date**, *valentine's day*⟩	datum	datum
⟨**date**, *asian pear*⟩	fruit	fruit
⟨**gold**, *stainless steel*⟩	**material**	metal
⟨**gold**, *chocolate brown*⟩	color	color
⟨**java**, *perl*⟩	language	language
⟨**mouse**, *mp3 player*⟩	device	device
⟨**mouse**, *prairie dog*⟩	**animal**	mammal
⟨**orange**, *red*⟩	color	color
⟨***rock***, *jazz*⟩	genre	genre
⟨**rock**, *stone*⟩	**material**	**material**
⟨**shell**, *bone*⟩	**material**	**material**
⟨**shell**, *exxon mobil corp.*⟩	company	company
⟨**spring**, *river*⟩	surface water	surface water
⟨**spring**, *summer*⟩	holiday/season	holiday/season
⟨**sun**, *wind power*⟩	renewable energy source	renewable energy source
⟨**sun**, *coca-cola*⟩	company	company
⟨**turkey**, *corned beef*⟩	**food**	meat
⟨**turkey**, *sierra leone*⟩	country	country
⟨*watch*, **cream**⟩	**good/product**	**good/product**
⟨*white*, **cream**⟩	color	color
⟨**jaguar**, *dog*⟩	**animal**	mammal
⟨**jaguar**, *bmw*⟩	carmakers /marque	carmakers /marque
⟨*sony*, **ge**⟩	company	company

In one dimension, Fig. 13 reports the performance of our RC and RCP approaches compared to our Basic approach without entity disambiguity. Experimental results show that the PCC value in RC is improved by 0.17 compared to Basic, and the PCC value in RCP is further improved by 0.28 compared to RC. These data reveal the effectiveness of RC and RCP with the concept clustering on the entity disambiguity. In the other dimension, Table 4 lists the dominant senses of 30 entity-entity pairs computed by RC and RCP approaches. In this table, we manually highlight the ambiguous entities and the false dominant senses in bold. From the experimental results, we can see that RC can correctly get the dominant senses of entity pairs in the calculation of pair similarity by 70 percent (namely 21/30), while it is up to 90 percent (namely 27/30) in RCP. Only three pairs (namely ⟨**rock**, *stone*⟩, ⟨**shell**, *bone*⟩ and ⟨*watch*, **cream**⟩) have the false dominant senses. This is because senses such as

TABLE 5
Example Pairs with Similarity Scores Computed by RCP Approach (H.R. = Human Rating, S.S. = Similarity Score)

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

'material' and 'good/product' are still too general for these given pairs. It is necessary to mention that we consider the dominant senses of 'animal' and 'food' are false in RC for the given pairs of ⟨**mouse**, *prairie dog*⟩, ⟨**jaguar**, *dog*⟩ and ⟨**turkey**, *corned beef*⟩. The reason is because 'animal' is more general than 'mammal' and 'food' is more general than 'meat' predicted in RCP. In addition, RCP aims to prune some intersectional non-dominant senses of the two terms in a pair. There is no common sense for the pair ⟨**apple**, *ipad*⟩ in RC, the dominant sense is hence "null" in both RC and RCP approaches.

5.5 Efficiency

In this section, we aim to observe the efficiency of our approaches. Fig. 16 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.

Fig. 17 reports the mean computation time on a pair of terms in our approaches compared to the other competitors using the box plot. 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 (IC) 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 takes about 60 times longer than RCP because it requires extracting lexico-syntactic patterns from snippets online.

Fig. 18 shows the mean computation time on different types of pairs using our approaches. RCP costs less than half the time of RC 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., "mammal" and "puppy" (with "dog" being the child of "mammal" and parent of "puppy"), we iteratively check the relations between every top ancestor concepts of an entity term and the concept term in RCP.

5.6 On Document Classification

In this section, we extend the above term similarity with the Rocchio classifier [28] in document classification. That is, we consider the similarity between terms simultaneously, when evaluating the similarity between a given testing document and the training documents in Rocchio. To test the effectiveness of the term similarity, we select four classification tasks from the benchmark database 20 Newsgroups [29] widely used in document classification. Meanwhile, we use the word-based VSM (Vector Space Model) and the term-based VSM to represent the documents respectively, where the feature weights are computed using the tfidf values. Table 6 shows the average F-Measure values of our Term-Similarity-based Rocchio algorithm compared to eight competing algorithms with different basic classifiers. From the experimental results, we can see that our Term-Similarity-based Rocchio algorithm leads the competition against the peers. These results reveal that introducing the term similarity indeed is beneficial to improve the prediction ability in document classification.

6 Related Work

In this section, we only discuss previous work on semantic similarity. Contrary to the semantic relatedness which represents the more general relationships such as part-whole and the co-occurrence, semantic similarity measures the

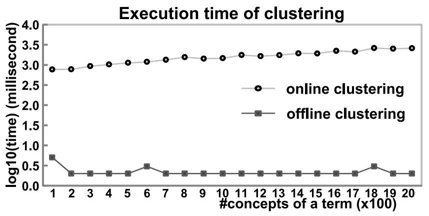

Fig. 16. Performance of RCP with online/offline clustering.

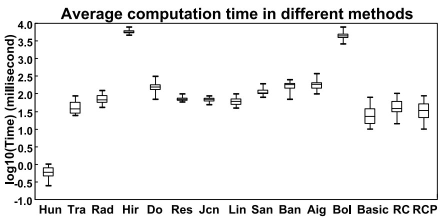

Fig. 17. Computation time in different approaches.

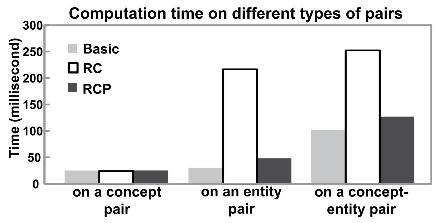

Fig. 18. Computation time on different types of pairs.

degree of taxonomic likeness between concepts and considers relations such as hyperonymy and synonymy, while most of them can be adapted or generalized to deal with semantic relatedness.

Semantic similarity measures are important in many web-related tasks, such as web information analysis [30] and query expansion [31]. 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 one computes similarity by the context of terms in large text corpora (such as the search snippets and web documents), that is, such similarities are derived from distributional properties of words or n-grams in the corpora. More details are as follows.

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 is the path-length based approach [5]. This path-length based approach is very simple, but has a low accuracy because it ignores the amount of information hidden in the concept nodes. More advanced approaches compute the similarity between terms by the Information Content of these terms with respect to the taxonomy structure, such as [6], [22], [23], [24], [32]. Meanwhile, some researchers also attempt to apply graph learning algorithms based on WordNet in the term similarity computation, such as a rooted weighted graph based algorithm [33], a WordNet-based personalized PageRank algorithm [7].

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* similarity function effectively picks the senses that are most suitable given two terms. All aforementioned methods cannot be easily adapted to use Probase because it is a general network, not a tree structure.

Corpus-based approach. All of methods in this direction use the statistics information hidden in the corpus to compute the similarity between terms. There are two categories according to the context representation, one is the distributional models and the other is the feature-based models for semantic representation. We call them as the distributional similarity methods and the feature-based similarity methods respectively.

Considering the distributional similarity methods, the representative works are below. Ido et al. proposed the probabilistic word association models based on the distributional word similarity [34] for the pseudo-word disambiguation. Toutanova et al. proposed a Markov chain model, whose stationary distribution is used to give word probability estimates [35]. Rohde et al. introduced a new vector-space method for deriving word-meanings from large corpora [36]. Mohammad and Hirst presented a framework to derive the distance between concepts from distributional measures of word co-occurrences [37]. Kazama et al. proposed a Bayesian method for robust distributional word similarities [38]. Piitulainen proposed a method based on syntactically determined co-occurrences and simple frequency weights to calculate a relatively large-scale similarity table of frequent nouns in a Finnish newspaper corpus [39].

The aforementioned methods aim to compute semantic similarity between terms more correctly, but the representations derived by distributional models are purely symbolic and are not grounded in perception and action [40]. Therefore, many feature-based models for semantic representation have been proposed as follows. Chen et al. proposed a double-checking model using the occurrences of terms in their search snippets to evaluate the semantic similarity [41]. Cilibrasi and Vitányi proposed a distance metric between words using only page-counts retrieved from a web search engine [42]. Bollegala et al. proposed a new similarity measure method using page counts and snippets from web search [26]. The above methods can easily get snippets and search results as the corpus, but they are time-consuming because i) snippets and search results must be obtained online; ii) it requires parsing of the returned text by the patterns. In addition, Radinsky et al. proposed a new model of temporal semantic analysis (TSA) to capture the temporal information of corpus [43]. In TSA, each concept is represented as time series over a corpus of temporally-ordered documents. It can improve the PCC, but it requires massive historical data.

TABLE 6
Classification on 20NewsGroups Data Sets (T.S. = Term Similarity, NB = Naive Bayes)

data set	Using BagOfWords				Using BagOfTerms				
	NB	SMO	J48	Rocchio	NB	SMO	J48	Rocchio	T.S.-based Rocchio
talk.politics.guns vs. sci.crypt	0.924	0.939	0.814	**0.962**	0.916	0.932	0.834	0.949	0.949
rec.autos vs. sci.med	0.870	0.925	0.767	0.946	0.891	0.936	0.748	0.965	**0.979**
rec.motorcycles vs. comp.graphics	0.954	0.963	0.886	0.948	0.966	0.977	0.910	0.968	**0.982**
talk.religion.misc vs. sci.space	0.895	0.944	0.838	0.938	0.901	0.928	0.842	0.961	**0.967**

In sum, the aforementioned 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. Meanwhile, corpus based methods are more suitable for specific languages.

7 Conclusions

We presented an efficient and effective approach for semantic similarity between terms with any multi-word expression. It uses an isA semantic network extracted from large Web corpus to provide contexts for the terms, employs a concept clustering algorithm to disambiguate the senses of the input terms, and finally applies a *Max* similarity function to compute the similarity. Extensive studies show that 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. In our future work, we will focus on how to apply our approach in short text categorization.

Acknowledgments

This work was supported in part by the Program for Changjiang Scholars and Innovative Research Team in University (PCSIRT) of the Ministry of Education, China, under grant IRT13059, the National 973 Program of China under grant 2013CB329604, the Specialized Research Fund for the Doctoral Program of Higher Education under grant 20130111110011, the Natural Science Foundation of China under grants (61273292, 61273297, 61229301, 61070131, 61305063), the Doctoral Fund of Ministry of Education of China under grant 2013JYBS0632, the China Postdoctoral Science Foundation under grant 2014M551801, MOE New Faculty under grant 20110073120023, and the Postdoctoral Science Foundation of Hefei University of Technology, China, under grant 2013HGBH0025. Peipei Li is the corresponding author. A preliminary version of this paper has been accepted for publication in the *Proceeding of ACM International Conference on Information and Knowledge Management* (*CIKM'13*), pp. 1401–1410, 2013.

References

[1] A. Budanitsky and G. Hirst, "Evaluating wordnet-based measures of lexical semantic relatedness," *Comput. Linguistics*, vol. 32, pp. 13–47, 2006.
[2] Y. Wang, H. Li, H. Wang, and K. Q. Zhu, "Concept-based web search," in *Proc. 31st Int. Conf. Conceptal Model. ER*, 2012, pp. 449–462.
[3] M. Gärtner, A. Rauber, and H. Berger, "Bridging structured and unstructured data via hybrid semantic search and interactive ontology-enhanced query formulation," *Knowl. Inf. Syst.*, vol. 41, pp. 761–792, 2014.
[4] G. A. Miller, "WordNet: A lexical database for english," *Commun. ACM*, vol. 38, no. 11, pp. 39–41, 1995.
[5] R. Rada, H. Mili, E. Bichnell, and M. Blettner, "Development and application of a metric on semanticnets," *IEEE Trans. Syst., Man Cybern.*, vol. 9, no. 1, pp. 17–30, Jan./Feb. 1989.
[6] P. Resnik, "Using information content to evaluate semantic similarity in a taxonomy," in *Proc. 14th Int. Joint Conf. Artif. Intell.*, 1995, pp. 448–453.
[7] E. Agirre, M. Cuadros, G. Rigau, and A. Soroa, "Exploring knowledge bases for similarity," in *Proc. 7th Int. Conf. Language Resources Eval.*, 2010, pp. 373–377.
[8] W. Wu, H. Li, H. Wang, and K. Q. Zhu, "Probase: A probabilistic taxonomy for text understanding," in *Proc. ACM SIGMOD Int. Conf. Manag. Data*, 2012, pp. 481–492.
[9] M. A. Hearst, "Automatic acquisition of hyponyms from large text corpora," in *Proc. 14th Conf. Comput. Linguistics*, 1992, pp. 539–545.
[10] Y. Song, H. Wang, Z. Wang, H. Li, and W. Chen, "Short text conceptualization using a probabilistic knowledgebase," in *Proc. 22nd Int. Joint Conf. Artif. Intell.*, 2011, pp. 2330–2336.
[11] A. W. Moore, "An intodductory tutorial on kd-trees," Computer Laboratory, University of Cambridge, Tech. Rep. 209, 1991.
[12] G. Miller and W. Charles, "Contextual correlates of semantic similarity," *Lang. Cogn. Process.*, vol. 6, pp. 1–28, 1998.
[13] H. Rubenstein and J. B. Goodenough, "Contextual correlates of synonymy," *Commun. ACM*, vol. 8, pp. 627–633, 1965.
[14] 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 *Proc. Human Language Technol.: Annu. Conf. North Am. Chapter Assoc. Comput. Linguistics*, 2009, pp. 19–27.
[15] (2002). [Online]. Available: http://www.cs.technion.ac.il/~gabr/resources/data/wordsim353/
[16] (2008). [Online]. Available: http://wn-similarity.sourceforge.net/
[17] (2003). [Online]. Available: http://en.wikipedia.org/wiki/Pearson_correlation_coefficient
[18] (1957). [Online]. Available: http://www.math.uwo.ca/mdawes/courses/344/kuhn-munkres.html
[19] (2010). [Online]. Available: http://www.codeproject.com/Articles/11835/Word-Net-based-semantic-similar ity-measurement
[20] G. Hirst and D. St-Onge, "Lexical chains as representations of context for the detection and correction of malapropisms," in *WordNet: An Electron. Lexical Database*. Cambridge, MA, USA: MIT Press, 1998, pp. 305–332.
[21] Q. Do, D. Roth, M. Sammons, Y. Tu, and V. Vydiswaran, "Robust, light-weight approaches to compute lexical similarity," University of Illinois, Champaign, IL, USA, Comput. Sci. Res. Tech. Rep., 2009.
[22] J. Jiang and D. Conrath, "Semantic similarity based on corpus statistics and lexical taxonomy," in *Proc. Int. Conf. Res. Comput. Linguistics*, 1997, pp. 19–33.
[23] D. Lin, "An information-theoretic definition of similarity," in *Proc. 15th Int. Conf. Mach. Learning*, 1998, pp. 296–304.
[24] D. Sánchez, M. Batet, and D. Isern, "Ontology-based information content computation," *Knowl.-Based Syst.*, vol. 24, pp. 297–303, 2011.
[25] S. Banerjee and T. Pedersen, "An adapted lesk algorithm for word sense disambiguation using wordnet," in *Proc. 3rd Int. Conf. Comput. Linguistics Intell. Text Process.*, 2002, pp. 136–145.
[26] D. Bollegala, Y. Matsuo, and M. Ishizuka, "A web search engine-based approach to measure semantic similarity between words," *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 7, pp. 977–990, Jul. 2011.
[27] M. Yoshida, M. Ikeda, S. Ono, I. Sato, and H. Nakagawa, "Person name disambiguation by bootstrapping," in *Proc. 33rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2010, pp. 10–17.
[28] (2009). [Online]. Available: http://en.wikipedia.org/wiki/Rocchio_algorithm#Algorithm
[29] (1999). [Online]. Available: http://kdd.ics.uci.edu/databases/20newsgroups/20newsgroups.html
[30] V. Evrim and D. McLeod, "Context-based information analysis for the web environment," *Knowl. Inf. Syst.*, vol. 38, pp. 109–140, 2014.
[31] C. Carpineto and G. Romano, "A survey of automatic query expansion in information retrieval," *ACM Comput. Surv.*, vol. 44, pp. 1–50, 2012.
[32] M. A. H. Taieb, M. B. Aouicha, and A. B. Hamadou, "A new semantic relatedness measurement using wordnet features," *Knowl. Inf. Syst.*, vol. 41, pp. 467–497, 2014.
[33] M. Alvarez and S. Lim, "A graph modeling of semantic similarity between words," in *Proc. Int. Conf. Semantic Comput.*, 2007, pp. 355–362.
[34] I. Dagan, L. Lee, and F. C. N. Pereira, "Similarity-based models of word cooccurrence probabilities," *Mach. Learn.*, vol. 34, pp. 43–69, 1999.
[35] K. Toutanova, C. D. Manning, and A. Y. Ng, "Learning random walk models for inducing word dependency distributions," in *Proc. Int. Conf. Mach. Learning*, 2004, pp. 103–110.

[36] D. L. T. Rohde, L. M. Gonnerman, and D. C. Plaut, "An improved model of semantic similarity based on lexical co-occurrence," *Commun. ACM*, vol. 8, pp. 627–633, 2005.

[37] S. Mohammad and G. Hirst, "Distributional measures of concept-distance: A task-oriented evaluation," in *Proc. Conf. Empirical Methods Natural Language Process.*, 2006, pp. 35–43.

[38] J. Kazama, S. D. Saeger, K. Kuroda, M. Murata, and K. Torisawa, "A Bayesian method for robust estimation of distributional similarities," in *Proc. 48th Annu. Meeting Assoc. Comput. Linguistics*, 2010, pp. 247–256.

[39] J. Piitulainen, *Explorations in the Distributional and Semantic Similarity of Words*. Helsinki, Finland: Helsinki Univ. Print, 2011.

[40] B. Riordan and M. N. Jones, "Redundancy in perceptual and linguistic experience: Comparing feature-based and distributional models of semantic representation," *Topics Cogn. Sci.*, vol. 3, pp. 303–345, 2011.

[41] H. Chen, M. Lin, and Y. Wei, "Novel association measures using web search with double checking," in *Proc. 21st Int. Conf. Comput. Linguistics 44th Annu. Meeting Assoc. Comput. Linguistics*, 2006, pp. 1009–1016.

[42] R. Cilibrasi and P. M. B. Vitányi, "The google similarity distance," *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 3, pp. 370–383, Mar. 2007.

[43] K. Radinsky, E. Agichtein, E. Gabrilovich, and S. Markovitch, "A word at a time: Computing word relatedness using temporal semantic analysis," in *Proc. 20th Int. Conf. World Wide Web*, 2011, pp. 337–346.

Peipei Li received the BS, MS, and PhD degrees from Hefei University of Technology in 2005, 2008, 2013, respectively. She is currently a postdoctoral researcher at Hefei University of Technology, China. She was a research fellow at Singapore Management University from 2008 to 2009. She was a student intern at Microsoft Research Asia between August 2011 CA, and December 2012. Her research interests are in data mining and knowledge engineering.

Haixun Wang received the BS and MS degrees in computer science from Shanghai Jiao Tong University in 1994 and 1996, respectively, and the PhD degree in computer science from the University of California in 2000. He joined Google Research, Mountain View, in 2013. From 2009 to 2013, he was with Microsoft Research, Asia, where he led the database team. From 2000 to 2009, he was with IBM Research, Watson. His research interests are in text analytics, semantic network and knowledge base, etc.

Kenny Q. Zhu graduated with the BEng degree in electrical engineering in 1999 and the PhD degree in computer science in 2005 from National University of Singapore. He is an associate professor and distinguished research professor in the Department of Computer Science and Engineering, Shanghai Jiao Tong University. From 2007 to 2009, he was a postdoctoral researcher at Princeton University. His research interest lies in the intersection of programming languages and databases.

Zhongyuan Wang received the BS and MS degrees in computer science from Renmin University in 2010 and 2007, respectively. He is currently working toward the PhD degree from Renmin University of China. He is a researcher at Microsoft Research Asia. After he graduated from RUC, he joined MSRA as a Research Software Development Engineer. His research interests include knowledge base, web data mining, machine learning, and natural language processing.

Xuegang Hu received the BS degree from the Department of Mathematics at Shandong University, and the MS and PhD degrees at Hefei University of Technology. He is a professor in the School of Computer Science and Information Engineering, Hefei University of Technology, China, and the director-general of Computer Association of Higher Education at Anhui Province. His research interests include data mining and knowledge engineering.

Xindong Wu received the PhD degree in artificial intelligence from the University of Edinburgh, Britain. He is a Yangtze River scholar in the School of Computer Science and Information Engineering, Hefei University of Technology, China, a professor of computer science at the University of Vermont, US. He is the Steering Committee chair of IEEE International Conference on Data Mining (ICDM) and editor-in-chief of *Knowledge and Information Systems* (*KAIS*). He was the editor-in-chief of *TKDE* from 2005 to 2008. His research interests include data mining and Big Data analytics. He is a fellow of the IEEE and the AAAS.