Expert Systems With Applications 000 (2015) 1–13

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Efficient link-based similarity search in web networks

Mingxi Zhang[a,b,*], Hao Hu[b], Zhenying He[b], Liping Gao[a,b], Liujie Sun[a]

[a] University of Shanghai for Science and Technology, 516 Jungong Road, Shanghai 200093, China
[b] Fudan University, 825 Zhangheng Road, Shanghai 201203, China

ARTICLE INFO

Keywords:
Similarity search
Web network
WebSim

ABSTRACT

Similarity search in web networks, aiming to find entities similar to the given entity, is one of the core tasks in network analysis. With the proliferation of web applications, including web search and recommendation system, SimRank has been a well-known measure for evaluating entity similarity in a network. However, the existing work computes SimRank iteratively over a huge similarity matrix, which is expensive in terms of time and space cost and cannot efficiently support similarity search over large networks. In this paper, we propose a link-based similarity search method, WebSim, towards efficiently finding similar entities in web networks. WebSim defines the similarity between entities as the 2-hop similarity of SimRank. To reduce computation cost, we divide the similarity search process into two stages: off-line stage and on-line stage. In the off-line stage, the 1-hop similarities are computed, and an optimized algorithm is designed to reduce the unnecessary accumulation operations on zero similarities. In the on-line stage, the 2-hop similarities are computed, and a pruning algorithm is developed to support fast query processing through searching similar entries from a partial sums index derived from the 1-hop similarities. The index items that are lower than a given threshold are skipped to reduce the searching space. Compared to the iterative SimRank computation, the time and space cost of similarity computation is significantly reduced, since WebSim maintains only the similarity matrix of 1-hop that is much smaller than that of multi-hop. Experiments through comparison with SimRank and its optimized algorithms demonstrate that WebSim has on average a 99.83% reduction in the time cost and a 92.12% reduction in the space cost of similarity computation, and achieves on average 99.98% NDCG.

© 2015 Elsevier Ltd. All rights reserved.

1. Introduction

Web networks are ubiquitous and can be found in many real applications, such as web social networks, e-mail networks and product co-purchasing networks. Similarity search in web networks aims to find the entities similar to a given entity. It is a special important task and draws extensive interests from various research fields, including recommendation systems, link prediction, approximate query processing and web search. For satisfying the requirements of above applications, some link-based similarity measures have been devoted in recent years, such as SimRank (Jeh & Widom, 2002), P-Rank (Zhao, Han, & Sun, 2009) and SimFusion (Xi et al., 2005). These methods compute similarities by exploring the indirect links among entities. The similarity between two entities is defined recursively with respect to a "random surfer" model. Compared with the text-based similarity measures (Ganesan, Garcia-Molina, & Widom, 2003), the link-based similarity measures produce systematically better correlation with human judgements (Maguitman, Menczer, Erdinc, Roinestad, & Vespignani, 2006), which provide a good way for effectively evaluating entity similarities.

Among the existing link-based similarity measures, SimRank (Jeh & Widom, 2002) can be regarded as one of the influential ones on account of the following reasons. First, SimRank is based on hyper-links and follows intuition that "two nodes are similar if they are referenced by similar nodes", which conforms to our basic understandings. Second, SimRank considers not only direct connections among nodes but also indirect connections, which helps discover valuable underlying relationships. Third, SimRank does not suffer from any field restrictions, which is applicable to any domain with entity-to-entity relationships.

However, the computation of SimRank is expensive in terms of time and space cost, which makes it less efficient for similarity search in large networks. The SimRank algorithm computes similarities through an iterative way, in which a huge similarity matrix need to be maintained for storing similarities among entities. This similarity matrix would become full after just a few iterations, and then the

* Corresponding author at: University of Shanghai for Science and Technology, 516 Jungong Road, Shanghai 200093, China. Tel.: +8615000751359.
E-mail addresses: mingxizhang10@fudan.edu.cn, waxl7461@aliyun.com (M. Zhang), huhao@fudan.edu.cn (H. Hu), zhenying@fudan.edu.cn (Z. He), lipinggao@fudan.edu.cn (L. Gao), liujiesunx@163.com (L. Sun).

http://dx.doi.org/10.1016/j.eswa.2015.07.042
0957-4174/© 2015 Elsevier Ltd. All rights reserved.

efficiency problem would be run into with network becoming massive. Although some techniques have been proposed for optimizing SimRank computation (Du, Li, Chen, Tan, & Zhang, 2015; Lee, Lakshmanan, & Yu, 2012; Lizorkin, Velikhov, Grinev, & Turdakov, 2010; Yu, Lin, & Zhang, 2014; Zhao, Xiao, Lin, Liu, & Zhang, 2013), they still need to maintain a large matrix for similarity computation, which are particularly inefficient in practice.

Motivated by above discussions, in this paper, we propose a link-based similarity search method, WebSim, towards efficiently finding similar entities in web networks. Our key observation is that the iterative computation of SimRank converges very fast, and there is little change in the returned rankings after the second iteration. Based on this, WebSim defines the similarity between entities as the SimRank similarity at the second iteration (2-hop similarity). The similarity search process is divided into two stages: off-line stage and on-line stage. In the off-line stage, we compute only the similarities at first iteration (1-hop similarities), which helps further reduce the pre-computation cost. In the on-line stage, we compute the 2-hop similarity between query and candidate based on the pre-computed 1-hop similarities. WebSim maintains only the similarity matrix of 1-hop that is much smaller than that of multi-hop, and therefore the expensive time and space cost for iterative SimRank computation would be decreased.

Our research faces the following two challenges. The first challenge is to reduce the computation cost in the off-line stage. Although the computation cost can be decreased to some extent by reducing the iteration number of similarity computation, however, it is inefficient in practice since many unnecessary accumulation operations on zero similarities are still involved, especially when the network grows large. For reducing the computation cost, we rewrite WebSim equation into a more efficient and simple form which is independent of initializing similarities. Based on this, we develop an optimized algorithm for efficiently computing 1-hop similarities, which decreases the computation cost by considering only the accumulation operations on non-zero similarities, and the unnecessary accumulation operations on zero similarities are not involved.

The second challenge is to reduce the execution time of on-line query processing. A straightforward approach is to search the entities similar to a given query over a pre-computed similarity matrix. Although this approach is extremely fast for on-line querying, however, the similarity computation is prohibitively expensive in terms of time and space cost (Jeh & Widom, 2002; Lizorkin et al., 2010; Zhao et al., 2013). On the other hand, if the similarities are not pre-computed, expensive operations would be involved to calculate the similarity between query and candidate, which increases the response time (He et al., 2014; Lee et al., 2012; Li, Liu, Yu, He, & Du, 2010b). For supporting fast on-line query processing, we develop a pruning algorithm which searches the similar entities over a designed partial sums index that is built off-line by utilizing the 1-hop similarities. For reducing the searching space, we skip the index items of lower partial sums by setting a threshold, which reduces the time cost of on-line similarity computation and prunes the candidates that are not promising. Experiments through comparison with SimRank and its optimized algorithms demonstrate that WebSim reduces the time and space cost of similarity computation by on average 99.83% and 92.12% respectively, and achieves on average 99.98% NDCG.

The remaining of this paper is organized as follows. Section 2 discusses the related work on similarity search. Section 3 gives some notations and SimRank overview. Section 4 gives WebSim equation. In Section 5, we give the WebSim pre-computation algorithm and the optimized techniques. In Section 6, we give the algorithms of websim-baseline and websim-pruning. Experimental studies are reported in Section 7. Finally, the conclusion are discussed in Section 8.

2. Related work

Due to the practical significance of similarity search in real networks, many approaches have been devoted in recent years. With respect to the focus of this paper, below we briefly describe the work on similarity search that is most relevant to the current work.

Some early approaches like Co-citation (Small, 1973), Bibliographic Coupling (Kessler, 1963) and Amsler (Amsler, 1972) measure similarity based on structural-context. Co-citation measures the similarity between two papers based on the common papers which cite both of them, formally, the similarity between paper *a* and paper *b* is defined as the number of papers which cite both *a* and *b*, while Bibliographic Coupling defines similarity as the number of papers cited by both *a* and *b*. Amsler fuses both Co-citation and Bibliographic Coupling for similarity computation. Akmal, Shih, and Batres (2014) developed a semantic similarity measure based on the comparison of classes in an ontology, in which the similarity between two classes is computed based on their attribute similarities. Liao and Xu (2015) employed cosine distance for computing the similarity between hesitant fuzzy linguistic elements. These approaches compute similarities based on 1-hop neighbors, the indirect connections are not considered for similarity computation, which would neglect some similar results when searching similar entities.

In recent years, some multi-hop neighbor-based similarity measures have been devoted. SimRank (Jeh & Widom, 2002) is one of these useful similarity measures for ranking nodes in order of relevance to a query node. The similarity between two nodes is defined as the expected distance for two random surfers to first meet at the same node when they walk along the network backwards. SimRank has been successfully employed in various applications, such as social recommender system (Li, Li, Chen, & Du, 2013), similarity join (Tao, Yu, & Li, 2014) and query aggregation (Xu, Li, Chen, & Sun, 2015). P-Rank (Zhao et al., 2009) enriches SimRank by jointly encoding both in- and out-link relationships into structural similarity computation. The intuition behind P-Rank is that “two nodes are similar if (1) they are referenced by similar nodes or (2) they reference similar nodes”. P-Rank is a general form of some other similarity measures, including Co-citation, Bibliographic Coupling and SimRank. E-Rank (Zhang, He, Hu, & Wang, 2012) measures similarity between social entities based on the intuition that “two entities are similar if they can arrive at common entities”, which exploits the meetings of any path length and relationship strength to enhance the effectiveness. The above similarity measures are mainly on the homogeneous networks that consist of the objects and links of same type, the heterogeneity of real networks is not addressed.

There are also some similarity measures on the heterogeneous networks that consist of the objects and links of different types. SimFusion (Xi et al., 2005) uses an unified relationship matrix (URM) to represent the heterogeneous web objects and the inter-relationships among these web objects. Typical web objects include products, e-mail users, web pages and the like. The similarity matrix of SimFusion is computed iteratively over URM, which helps overcome the data sparseness problem and detect the latent relationships among heterogeneous data objects. PathSim (Sun, Han, Yan, Yu, & Wu, 2011) is a meta path-based similarity measure, which allows users provide a meta path to measure similarities from different perspectives. However, it is difficult for the users to choose an appropriate meta path especially when the network schema is unknown.

The above methods consider multi-hop neighbors for similarity computation, which can find more comprehensive results compared to the 1-hop neighbor-based measures. However, the iterative computation of these methods is expensive, in which a huge matrix need to be maintained for storing the multi-hop similarities. With iteration increasing, this matrix would become full gradually, which decreases the efficiency of similarity computation.

Some optimization techniques on similarity computation have been developed recently. Lizorkin et al. (2010) optimized SimRank by essential node pairs, partial sums and threshold-sieved, which reduces the computational complexity from $O(n^2d^2)$ to $O(n^2d)$, and further reduces to $\min(O(n^2d), O(n^3/\log_2 n))$ by using cross summation, where n is the node number of a given graph and d is the average degree. Li et al. (2010a) introduced a non-iterative SimRank computation method in dynamic networks, which rewrites SimRank into a non-iterative form based on the Kronecker product and vectorization operators. Yu, Lin, Zhang, Chang, and Pei (2013) proposed SimRank* for resolving the counter-intuitive zero-similarity issues, which speeds up the computation procedure by an induced graph. Zhao et al. (2013) proposed a partition-based approach to tackle the efficiency problem of SimRank by dividing the data graphs into variable-size non-overlapping partitions.

Yu et al. (2014) proposed an incremental SimRank computation algorithm, which characterizes the SimRank update matrix ΔS w.r.t. every link update via a rank-one Sylvester matrix equation. The "affected areas" of ΔS is identified by an effective pruning strategy, in which the unnecessary similarity re-computation is skipped. Yu, Lin, Zhang, and McCann (2015) employed a clustering strategy to eliminate duplicate computations occurring in partial sums, the SimRank matrix is represented as an exponential sum of transition matrices, which leads to a further speedup in the convergence rate of SimRank iterations. Du et al. (2015) extended SimRank to define a similarity measure on probabilistic graphs and dynamic graphs, in which the probabilistic transition matrix is computed over uncertain graphs by a designed SubG algorithm that utilizes multiple sub-graphs for computing transition probability by dynamic programming methods.

Nevertheless, although the above optimization techniques can reduce the computation cost in some degree, they still need to maintain a huge matrix to store the multi-hop similarities, which are particularly inefficient in practice and cannot be efficiently applied to large networks.

To reduce the pre-computation cost, NetSim (Zhang, Hu, He, & Wang, 2015) computes the object similarities on-line by combining the pre-computed attribute similarities. However, the attribute similarities are computed in an iterative mode, which is less efficient when network becomes large. Besides, this method cannot handle the networks beyond x-star network schema. Iterative Single-Pair SimRank (ISP) (He et al., 2014) computes the SimRank similarity score for a single pair of nodes in a graph, in which a new data structure, position matrix, is used for facilitating computation of the first-meeting probabilities of two random surfers. The time cost of ISP is always less than the original algorithm SimRank, since the large similarity matrix is not required. However, the expensive operations of on-line similarity computation would increase the time cost of query processing.

Compared to the above methods, WebSim computes the actual similarities at the second iteration on-line according to the pre-computed 1-hop similarities, which saves the expensive operations for iterative similarity computation. As well as other similarity measures that compute similarities based on multi-hop neighbors, WebSim can effectively capture the latent similar entities that have common multi-hop neighbors but no common 1-hop neighbors even though it considers only 2-hop neighbors, since only a small portion of entities are similar in real networks (Yin, Han, & Yu, 2006).

3. Preliminaries

3.1. Notations

Before we discuss further on similarity search, we first list the notations for the subsequent discussions.

Definition 1 (Web Network). A web network is defined as a directed graph $G(V, E)$, where V is a set of nodes and E is a set of edges. A directed edge $e(u, v) \in E$ represents a link from u to v, where $u, v \in V$ represent entities of web network.

The in-neighbor set of node v is denoted by $I(v)$, which represents the set of entities cited by v; and the out-neighbor set of node v is denoted by $O(v)$, which represents the set of entities cite v. Symbol $I_i(v)$ denotes the ith neighbor in set $I(v)$, and $O_i(v)$ denotes the ith neighbor in set $O(v)$.

Definition 2 (l-hop in-neighbor). A node $x \in V$ is a l-hop in-neighbor of $v \in V$ if and only if there exists a path of length l from x to v, which is also a reverse path of length l from v to x.

According to the definition of l-hop in-neighbor, we give the definition of 2-hop in-neighbor, i.e., the in-neighbor's in-neighbor. We can also give the definition of l-hop out-neighbor similarly.

3.2. SimRank overview

In this subsection, we give a brief review of SimRank. For nodes $a, b \in V$, the SimRank similarity between them is denoted by $s(a, b) \in [0, 1]$, which is defined as $s(a, b) = 1$ if $a = b$, otherwise:

$$s(a,b) = \frac{C}{|I(a)||I(b)|} \sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(b)|} s(I_i(a), I_j(b)) \tag{1}$$

where $C \in (0, 1)$ is the decay factor. For preventing division by zero in Eq. (1), $s(a, b)$ is defined as zero when $|I(a)| = 0$ or $|I(b)| = 0$.

We will further refer to $s(*, *)$ as the theoretical similarity of SimRank, and refer to $s(a, b)$ as the theoretical similarity between a and b. A solution to compute SimRank similarities is reached by iteration to a fixed-point. At iteration l, the computational similarity between a and b is denoted by $R_l(a, b)$. The iterative computation is started with $R_0(*, *)$, which is defined as:

$$R_0(a,b) = \begin{cases} 0, & \text{if } a \neq b \\ 1, & \text{if } a = b \end{cases} \tag{2}$$

When $l \neq 0$, $R_l(a, b)$ is defined as $R_l(a, b) = 1$ if $a = b$, otherwise:

$$R_l(a,b) = \frac{C}{|I(a)||I(b)|} \sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(b)|} R_{l-1}(I_i(a), I_j(b)) \tag{3}$$

The time cost for computing the similarities of all node pairs at the lth iteration is $O(ld^2n^2)$, and the space cost is $O(n^2)$, where $d = |E|/|V|$ is the average degree and n is the node number of a given graph.

Eq. (1) is written for every pair of nodes $a, b \in V$, resulting in a set of n^2 equations for a network of size n. As mentioned by Jeh and Widom (2002), the solution to n^2 iterative SimRank equations always exists and is unique, which can be reached by iterative computation to a fixed point. The solution to iterative SimRank converges to a limit, i.e., $\forall a, b \in V$, $\lim_{l \to \infty} R_l(a, b) = s(a, b)$, which satisfies the recursive SimRank equation. In real applications, iterative SimRank converges very fast. Empirically, the iteration number is set as $l = 5$ to derive SimRank for all pairs of nodes in real networks.

Theorem 1. *For entities $a, b \in V$, decay factor $C \in (0, 1)$ and iteration l, we have $R_l(a, b) = R_l(b, a)$.*

Theorem 1 demonstrates the symmetry property of SimRank which has been proved in Jeh and Widom (2002).

Theorem 2. *For entities $a, b \in V$, decay factor $C \in (0, 1)$ and iteration l, the average upper bound of the difference between $s(a, b)$ and $R_l(a, b)$ is $\mathrm{UB}(\mathrm{SimRank@}l) = s(a, b) - R_l(a, b) = (\frac{C}{d})^{l+1}$, where SimRank@$l$ is the SimRank at iteration l.*

Theorem 2 gives the average upper bound of the difference between the theoretical and computational similarity scores of SimRank, the proof can be found in (Lee et al., 2012).

4. WebSim equation

The similarity between web entities can be characterized by their linked neighbors based on the intuition that "similar entities are usually linked with similar entities", which conforms to the intuition of SimRank. Therefore, SimRank can be regarded as a promising measure for evaluating similarity between entities. However, the real networks are typically large, and the efficiency problem would be run into when applying SimRank to large networks.

As mentioned by Jeh and Widom (2002), the relative rankings would become stabilizing within 5 iterations, and there is little change on returned rankings after the second iteration. Therefore, we can reduce the computation cost by restricting the iteration number of similarity computation.

The similarity $R_l(a, b)$ between a and b can also be referred to as l-hop similarity. WeSim similarity is defined as the 2-hop similarity of SimRank. The WebSim similarities are divided into two stages: off-line stage and on-line stage. In the off-line stage, we compute only the 1-hop similarities, specifically, the 1-hop similarity between a and b is defined as $R_1(a, b) = 1$ if $a = b$, otherwise:

$$R_1(a,b) = \frac{C}{|I(a)||I(b)|} \sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(b)|} R_0(I_i(a), I_j(b)) \tag{4}$$

In the on-line stage, we compute the 2-hop similarity based on 1-hop similarities. The 2-hop similarity between a and b is defined as $R_2(a, b) = 1$ if $a = b$, otherwise:

$$R_2(a,b) = \frac{C}{|I(a)||I(b)|} \sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(b)|} R_1(I_i(a), I_j(b)) \tag{5}$$

By reducing the iteration number of SimRank computation, the time and space cost for iterative similarity computation can be reduced. In the off-line stage, only the 1-hop similarities are computed, which further saves the pre-computation cost.

Corollary 1. *For entities $a, b \in V$, decay factor $C \in (0, 1)$, the average upper bound of the difference between $s(a, b)$ and $R_2(a, b)$ is* $\mathrm{UB}(\mathrm{WebSim}) = s(a,b) - R_2(a,b) = (\frac{C}{d})^3$.

Proof. By Theorem 2, we have $\mathrm{UB}(\mathrm{SimRank@}l) = s(a,b) - R_l(a,b) = (\frac{c}{d})^{l+1}$, replace l by 2, we get $\mathrm{UB}(\mathrm{WebSim}) = s(a,b) - R_2(a,b) = (\frac{c}{d})^3$. □

Corollary 1 gives the average upper bound of the difference between the theoretical and computational similarity scores of WebSim.

Corollary 2. *For entities $a, b \in V$, decay factor $C \in (0, 1)$ and iteration $l \geq 2$, we have* $\mathrm{UB}(\mathrm{SimRank@}l) - \mathrm{UB}(\mathrm{WebSim}) = (\frac{C}{d})^3 - (\frac{C}{d})^{l+1}$.

Corollary 2 gives the average upper bound of the difference between the computational similarity scores of SimRank@l and WebSim, which can be directly derived from Theorem 2 and Corollary 1.

5. Off-line similarity computation

5.1. Naive pre-computation algorithm

For a given web network G, the naive procedure for computing 1-hop similarity matrix is illustrated in Algorithm 1. From this algorithm, we can derive that the average time cost for computing the 1-hop similarity between two entities is $O(d^2)$, and then the time cost for computing 1-hop similarity matrix can derived as $O(d^2n^2)$. For any $a \in V$, we have $R_1(a, b) \neq 0$ if $x \in I(a) \wedge b \in O(x)$, and therefore the maximal number of non-zero entries in the row vector of entity a is $\sum_{x \in I(a)} |O(x)|$. So the maximal average number of non-zero entries in the row vectors of R_1 can be further derived as d^2. Thus, the space complexity of 1-hop similarity matrix is $O(d^2 n)$.

Algorithm 1 Naive algorithm for computing 1-hop similarity matrix.

```
Input:
    Network $G(V, E)$;
Output:
    WebSim score $R_1(a, b), \forall a, b \in V$;
 1: Initialize $R_0(a, b), \forall a, b \in V$ by Eq. (??);
 2: For $a \in V$ do
 3:   For $b \in V$ do
 4:     if $a = b$ then
 5:       $R_1(a, b) \leftarrow 1$;
 6:     else
 7:       $R_1(a, b) \leftarrow \frac{C}{|I(a)||I(b)|} \sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(b)|} R_0(I_i(a), I_j(b))$;
 8:     end if
 9:   end for
10: end for
```

5.2. Optimized pre-computation algorithm

Compared to the iterative SimRank computation, the naive pre-computation algorithm can reduce the time and space cost in the off-line stage. However, with network becoming large, this algorithm would suffer from some limitations on computation efficiency. First, this algorithm needs to compute the similarity scores of all node pairs even only a small portion of them is non-zero. Second, when computing the similarity of a node pair, all the initialized similarity scores need to be checked, i.e., it cannot skip the zero similarity scores, which wastes a lot of time cost. Therefore, the naive pre-computation algorithm is particularly inefficient in practice. Accordingly, we next introduce an optimized algorithm to reduce the time cost for computing 1-hop similarities.

By Eq. (2), we have $R_0(I_i(a), I_j(b)) = 0$ if $I_i(a) \neq I_j(b)$, which gives no contribution to $R_1(a, b)$; and $R_0(I_i(a), I_j(b)) = 1$ if $I_i(a) = I_j(b)$, only this case is accounted for computing $R_1(a, b)$. And then, we consider only the meetings that two surfers start from a and b when they respectively walk along paths $x \to a$ and $x \to b$ backwards and meet at x , where $x \in I(a) \cap I(b)$. Thus, Eq. (4) can be equivalently rewritten as:

$$R_1(a,b) = \frac{C}{|I(a)||I(b)|} \sum_{x \in I(a) \cap I(b)} R_0(x, x) \tag{6}$$

By Eq. (2), we get $R_0(x, x) = 1$, and then Eq. (6) can be formalized as:

$$R_1(a,b) = \frac{C|I(a) \cap I(b)|}{|I(a)||I(b)|} \tag{7}$$

By Eq. (7), we have $R_1(a, b) = 0$ if $I(a) \cap I(b) = \varnothing$, in this case, the operations of similarity computation are unnecessary. On the other hand, if $I(a) \cap I(b) \neq \varnothing$, we have $R_1(a, b) \neq 0$, which can be accounted for computing $R_1(a, b)$.

Based on the above discussions, we give an optimized algorithm for computing 1-hop similarity matrix, which is shown in Algorithm 2. For each $a \in V$, line 3 to 10 captures each node $b \in O(x)$ for $x \in I(a)$ that connects with a through a symmetrical path $a \leftarrow x \rightarrow b$, and updates $R_1(a, b)$ by accumulating $\frac{C}{|I(a)||I(b)|}$, which skips the unnecessary accumulation operations on zero similarity scores.

The time complexity of this algorithm is derived as $O(d^2 n)$. As mentioned by Faloutsos, Faloutsos, and Faloutsos (1999), the web network follows a power-law degree distribution, the probability that a node has θ in- and out-links, $p(\theta)$, is on the order θ^{-a}, where a is a degree exponent. And then we can get that the expected value of degree is $E(\theta) = a/(a-1)$. In this case, the time complexity of this algorithm is $O(a^2 n/(a-1)^2)$. Similarly, the space complexity of the 1-hop similarity matrix can be derived as $O(a^2 n/(a-1)^2)$, which is very sparse compared to a full similarity matrix of $n \times n$.

Algorithm 2 Optimized algorithm for computing 1-hop similarity matrix.

```
Input:
    Network $G(V, E)$;
Output:
    WebSim score $R_1(a, b), \forall a, b \in V$;
1:  For $a \in V$ do
2:      $R_1(a, a) \leftarrow 1$;
3:      For $x \in I(a)$ do
4:          For $b \in O(x)$ do
5:              if $a = b$ then
6:                  Continue;
7:              end if
8:              $R_1(a, b) \leftarrow R_1(a, b) + \frac{C}{|I(a)||I(b)|}$;
9:          end for
10:     end for
11: end for
```

6. On-line query processing

6.1. Top-k similarity search under WebSim

Under the definition of WebSim, we now give the definition of top-k similarity search problem, which is described as follows.

Definition 3. (Top-k similarity search under WebSim). In a web network $G = (V, E)$, the top-k similarity search for a given entity q is to find k most similar entities ranked with similarities descending, such that $R_2(q, a) \geq R_2(q, a')$ for a in the returning list and a' not, where R_2 is the similarity function under WebSim.

6.2. WebSim-baseline

For a given query q, the baseline method for finding top-k similar entities is that: (1) for each candidate $a \in V$, we compute the similarity between q and a using Eq. (5), and add a to a candidate set C if $|C| < k$, otherwise: we remove $a' \in$ C corresponds to the lowest similarity from C if $R_2(q, a') < R_2(q, a)$, and add a to C; and (2) we sort the k candidates in C and return them. The time cost for finding top-k similar entities consists of two parts: one part is the time cost for selecting k entities from the n unsorted entities, which is derived as $O((d^2 + k)n)$; and another part is the time cost for sorting these k entities, denoted by $O(\square(k))$, which is depends on the sort algorithm. Therefore, the total time cost for finding top-k similar entities is $O((d^2 + k)n + \square(k))$. Compared to SimRank, although WebSim requires less computation cost in off-line stage, however, the time cost of on-line query processing would be significantly increased, since the similarity between query and candidate is computed on-line, while SimRank computes the whole similarity matrix in the off-line stage.

6.3. Optimized on-line query processing algorithm

In WebSim-baseline algorithm, two factors that increase the computation cost of query processing are involved. First, the more candidates to check, the more time the algorithm will take; and second, the more operations for computing the similarity between query and candidate, the more time will take. Therefore, the intuition to speed up the search process is to reduce the unnecessary operations for similarity computation and prune the unpromising candidates of being checked.

6.3.1. Rewrite WebSim similarity equation

For reducing the unnecessary operations for computing similarity between query and candidate, we next introduce an optimization

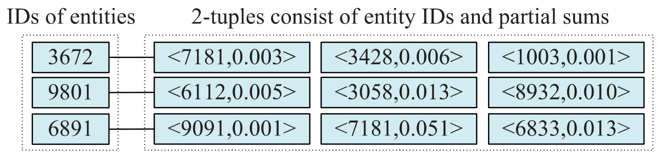


Fig. 1. A Fraction of partial sums index.

technique based on partial sums (Lizorkin et al., 2010), which allows reducing access operations to $R_1(*, *)$ that is required for computing $R_2(*, *)$. For given $I(a)$ and $I_j(b) \in I(b)$, the partial sums corresponds to R_1 is defined as:

$$\text{partial}_{I(a)}^{R_1}(I_j(b)) = \sum_{i=1}^{|I(a)|} R_1(I_i(a), I_j(b)) \tag{8}$$

The partial sums are computed in the off-line stage, based on which the similarity between entities can be computed on-line. Specifically, $R_2(a, b)$ is defined as $R_2(a, b) = 1$ if $a = b$, otherwise:

$$R_2(a, b) = \frac{C}{|I(a)||I(b)|} \sum_{j=1}^{|I(b)|} \text{partial}_{I(a)}^{R_1}(I_j(b)) \tag{9}$$

In practice, not all the partial sums can give enough contribution to the similarity scores, and some lower partial sums can be skipped when computing similarities. Next we skip the partial sums of lower value by setting a threshold ε. The partial sums $\text{partial}_{I(a)}^{R_1}(I_j(b))$ under threshold ε is denoted by $\text{partial}_{I(a)}^{R_1,\varepsilon}(I_j(b))$, which is defined as:

$$\text{partial}_{I(a)}^{R_1,\varepsilon}(I_j(b)) = \sum_{i=1}^{|I(a)|} R_1(I_i(a), I_j(b)) \tag{10}$$

if right-hand side of Eq. (10) is bigger than ε, otherwise $\text{partial}_{I(a)}^{R_1,\varepsilon}(I_j(b)) = 0$. And then $R_2(a, b)$ can be defined approximately as $R_2^{\varepsilon}(a, b) = 1$ if $a = b$, otherwise:

$$R_2^{\varepsilon}(a, b) = \frac{C}{|I(a)||I(b)|} \sum_{j=1}^{|I(b)|} \text{partial}_{I(a)}^{R_1,\varepsilon}(I_j(b)) \tag{11}$$

By Eq. (11), the time cost for computing similarity between query and each candidate can be derived as $O(d)$, so the time cost of WebSim-baseline can be reduced to $O((d + k)n + \square(k))$, which is significantly lower than WebSim-baseline.

6.3.2. Partial sums index

In practise, the entities similar to a given query are not too many, some entities corresponds to lower similarities can be skipped when processing queries. For a given query q, the operation for computing similarity $R_2^{\varepsilon}(q, x)$ between q and candidate x is unnecessary if $R_2^{\varepsilon}(q, x)$ is zero or closed to zero, and would increase the response time of on-line query processing.

For improving the efficiency of on-line query processing, we introduce an index structure, called partial sums index, for storing partial sums. The partial sums index is formalized as $P^{\varepsilon} = \{P^{\varepsilon}(y) | y \in V \wedge O(y) \neq \emptyset\}$ corresponds to the whole partial sums index, where $P^{\varepsilon}(y) = \{\langle a, \text{partial}_{I(a)}^{R_1,\varepsilon}(y)\rangle | \text{partial}_{I(a)}^{R_1,\varepsilon}(y) \neq 0\}$ is a set of 2-tuples corresponds to $\langle$nodeID, partial sums$\rangle$. An example of partial sums index is shown as Fig. 1, where the 2-tuples consist of entity IDs and partial sums. For example, $\langle 7181, 0.003\rangle$ corresponds to "3672" means the value of partial sums $\text{partial}_{I(7181)}^{R_1,\varepsilon}(3672)$ is 0.003.

The procedure for building partial sums index is shown as Algorithm 3. In line 4 to 12, for each node $a \in V$, the partial sums $\text{partial}_{I(a)}^{R_1,\varepsilon}(y)$ corresponds to node $y \in \{y | y \in V \wedge R_1(x, y) > 0\}$ is updated by accumulating $R_1(x, y)$ for each $x \in I(a)$. Here, B is a temporary

ARTICLE IN PRESS

Algorithm 3 Index building.

```
Input:
    Network $G(V, E)$, 1-hop similarity matrices $R_1$, threshold $\varepsilon$;
Output:
    Partial sums index $P^{\varepsilon}$;
 1: Initialize $P^{\varepsilon}$ as $\emptyset$;
 2: For $a \in V$ do
 3:   Initialize set $\mathsf{B}$ as $\emptyset$;
 4:   For $x \in I(a)$ do
 5:     For $y \in \{y | y \in V \wedge R_1(x, y) > 0\}$ do
 6:       if $\mathrm{partial}_{I(a)}^{R_1,\varepsilon}(y) \in \mathsf{B}$ then
 7:         $\mathrm{partial}_{I(a)}^{R_1,\varepsilon}(y) \leftarrow \mathrm{partial}_{I(a)}^{R_1,\varepsilon}(y) + R_1(x, y)$;
 8:       else
 9:         $\mathrm{partial}_{I(a)}^{R_1,\varepsilon}(y) \leftarrow R_1(x, y)$, $\mathsf{B} \leftarrow \mathsf{B} \cup \{\mathrm{partial}_{I(a)}^{R_1,\varepsilon}(y)\}$;
10:       end if
11:     end for
12:   end for
13:   For $\mathrm{partial}_{I(a)}^{R_1,\varepsilon}(y) \in \mathsf{B}$ do
14:     if $\mathrm{partial}_{I(a)}^{R_1,\varepsilon}(y) > \varepsilon$ then
15:       if $P^{\varepsilon}(y) \in P^{\varepsilon}$ then
16:         $P^{\varepsilon}(y) \leftarrow P^{\varepsilon}(y) \cup \langle a, \mathrm{partial}_{I(a)}^{R_1,\varepsilon}(y)\rangle$;
17:       else
18:         $P^{\varepsilon}(y) \leftarrow P^{\varepsilon}(y) \cup \langle a, \mathrm{partial}_{I(a)}^{R_1,\varepsilon}(y)\rangle$, $P^{\varepsilon} \leftarrow P^{\varepsilon} \cup P^{\varepsilon}(y)$;
19:       end if
20:     end if
21:   end for
22: end for
```

set consists of partial sums. In this step, only node $y \in \{y | y \in V \wedge R_1(x, y) > 0\}$ is considered for computing partial sums $\mathrm{partial}_{I(a)}^{R_1,\varepsilon}(y)$. In line 13 to 21, we add $\langle a, \mathrm{partial}_{I(a)}^{R_1,\varepsilon}(y)\rangle$ to partial sums index P^{ε} if $\mathrm{partial}_{I(a)}^{R_1,\varepsilon}(y) > \varepsilon$. The idea of this algorithm is similar to the accumulation operations for computing matrix R_1 as discussed before.

The time complexity of this algorithm can be easily derived as $O(dnn')$, where $n' \le n$ is the average number of non-zero entries in the row vectors of matrix R_1. As mentioned in Section 5.1, the maximal average number of non-zero entries in the row vectors of matrix R_1 is d^2, therefore, the time complexity of index building is $O(d^3n)$.

6.3.3. *WebSim-pruning*

Based on partial sums index, we next introduce an optimized on-line query processing algorithm, called WebSim-pruning, which utilizes both partial sums index and accumulation operations for speeding up the on-line query processing.

The procedure of WebSim-pruning is shown in Algorithm 4, which searches the top-k most similar entities to a given query q over a partial sums index P^{ε}. In line 1, we initialize $\mathsf{Q}(\mathsf{C}, \mathsf{S})$ by setting both C and S as $\emptyset$, where C is a set of candidates, and S is a similarity set corresponds to the candidates. Line 2–12 updates the similarity between q and a by accumulating $\frac{C}{|I(q)||I(y)|}\mathrm{partial}_{I(q)}^{R_1,\varepsilon}(I_j(y))$. For $y \in I(q)$, we first get each candidate a such that $\langle a, \mathrm{partial}_{I(a)}^{R_1,\varepsilon}(y)\rangle \in P^{\varepsilon}(y)$, and update $\mathsf{S}(a)$ by accumulating $\frac{C}{|I(q)||I(y)|}\mathrm{partial}_{I(q)}^{R_1,\varepsilon}(I_j(y))$ if $a \in \mathsf{C}$, otherwise we need to add a new element $\mathsf{S}(a)$ to S. In this step, only the non-zero partial sums are accounted for similarity computation. Function GetSortedEntity(k, Q) is used for getting the k most similar entities from Q, the basic process is that first get the k most similar entities from C according to their corresponding similarities in S, then sort and return them. The time cost of this algorithm can be derived as $O(\sum_{y \in I(q)} l_{\varepsilon}(y) + k|\mathsf{C}| + \square(k))$, where $l_{\varepsilon}(y)$ is the size of $P^{\varepsilon}(y)$.

Algorithm 4 WebSim-pruning algorithm.

```
Input:
    Network $G(V, E)$, partial sums index $P^{\varepsilon}$, query $q$ and parameter $k$;
Output:
    Top-$k$ most similar sorted web entities;
 1: Initialize $\mathsf{Q}(\mathsf{C}, \mathsf{S})$ by setting $\mathsf{C}$ and $\mathsf{S}$ as $\emptyset$;
 2: For $y \in I(q)$ do
 3:   For $\langle a, \mathrm{partial}_{I(a)}^{R_1,\varepsilon}(y)\rangle \in P^{\varepsilon}(y)$ do
 4:     if $a \in \mathsf{C}$ then
 5:       $\mathsf{S}(a) \leftarrow \mathsf{S}(a) + \frac{C}{|I(q)||I(y)|}\mathrm{partial}_{I(q)}^{R_1,\varepsilon}(I_j(y))$;
 6:     else
 7:       $\mathsf{S}(a) \leftarrow \frac{C}{|I(q)||I(y)|}\mathrm{partial}_{I(q)}^{R_1,\varepsilon}(I_j(y))$;
 8:       $\mathsf{C} \leftarrow \mathsf{C} \cup \{a\}$;
 9:       $\mathsf{S} \leftarrow \mathsf{S} \cup \{\mathsf{S}(a)\}$;
10:     end if
11:   end for
12: end for
13: return GetSortedEntity($k$, $\mathsf{Q}$);
```

From this algorithm, we get that the candidate set corresponds to node $y \in I(q)$ is $\mathsf{C}(y) = \{a | \langle a, \mathrm{partial}_{I(a)}^{R_1,\varepsilon}(y)\rangle \in P^{\varepsilon}(y)\}$. Therefore, the whole candidate set corresponds to query q is $\mathsf{C} = \cup_{y \in I(q)}\mathsf{C}(y) = \cup_{y \in I(q)}\{a | \langle a, \mathrm{partial}_{I(a)}^{R_1,\varepsilon}(y)\rangle \in P^{\varepsilon}(y)\}$. When increasing ε, the size of $\mathsf{C}(y)$ would have a downward trend, and hence the size of C would have a downward trend as well. Therefore, the time cost for choosing the k entities from C would be decreased. Note that the size of $\mathsf{C}(y)$ is equal to the size of $P^{\varepsilon}(y)$.

Compared to the pruning SimRank algorithm (Jeh & Widom, 2002), WebSim-pruning reduces the computation cost from the following two aspects. On the one hand, WebSim-pruning can skip the accumulation operations on zero similarities during similarity computation, while the pruning SimRank algorithm needs to compute the similarities of all node pairs within a radius r, which involves lots of zero similarities. On the other hand, WebSim-pruning can reduce the number of candidates by removing items lower than ε from partial sums index, while the pruning SimRank needs to check all the candidates of lower similarities.

Theorem 3. *For entities $a, b \in V$, decay factor $C \in (0, 1)$ and threshold ε, we have* $0 \le R_2(a, b) - R_2^{\varepsilon}(a, b) \le \frac{C \cdot \varepsilon}{|I(b)|}$.

Proof. By Eqs. (9) and (11), we can derive

$$R_2(a, b) - R_2^{\varepsilon}(a, b) = \frac{C}{|I(a)||I(b)|} \times \sum_{i=1}^{|I(a)|} \left(\mathrm{partial}_{I(a)}^{R_1}(I_j(b)) - \mathrm{partial}_{I(a)}^{R_1,\varepsilon}(I_j(b))\right).$$

By Eqs. (8) and (10), we get $\mathrm{partial}_{I(a)}^{R_1}(I_j(b)) - \mathrm{partial}_{I(a)}^{R_1,\varepsilon}(I_j(b)) = 0$ if $\mathrm{partial}_{I(a)}^{R_1}(I_j(b)) > \varepsilon$, otherwise $0 < \mathrm{partial}_{I(a)}^{R_1}(I_j(b)) - \mathrm{partial}_{I(a)}^{R_1,\varepsilon}(I_j(b)) \le \varepsilon$, and then we can derive $0 \le \mathrm{partial}_{I(a)}^{R_1}(I_j(b)) - \mathrm{partial}_{I(a)}^{R_1,\varepsilon}(I_j(b)) \le \varepsilon$, and consequently we have

$$\begin{aligned}\frac{C}{|I(a)||I(b)|}\sum_{i=1}^{|I(a)|} 0 \le \frac{C}{|I(a)||I(b)|}\sum_{i=1}^{|I(a)|} & \left(\mathrm{partial}_{I(a)}^{R_1}(I_j(b))\right. \\ & \left.-\mathrm{partial}_{I(a)}^{R_1,\varepsilon}(I_j(b))\right) = R_2(a, b) - R_2^{\varepsilon}(a, b) \\ \le \frac{C}{|I(a)||I(b)|}\sum_{i=1}^{|I(a)|} \varepsilon = \frac{C \cdot \varepsilon}{|I(b)|},\end{aligned}$$

which gives $0 \le R_2(a, b) - R_2^{\varepsilon}(a, b) \le \frac{C \cdot \varepsilon}{|I(b)|}$. □

Theorem 3 gives the maximal difference between WebSim-baseline and WebSim-pruning, which is the maximal accuracy loss of WebSim-pruning. By setting $\varepsilon = 0$, we have $R_2(a,b) = R_2^{\varepsilon}(a,b)$, in this case, the maximal accuracy loss is 0.

Theorem 4. *For entities $a, b \in V$, decay factor $C \in (0,1)$ and threshold ε, we have $|R_2^{\varepsilon}(a,b) - R_2^{\varepsilon}(b,a)| \le \frac{C\cdot\varepsilon}{|I(b)|}$.*

Proof. By Theorem 3, we have $0 \le R_2(a,b) - R_2^{\varepsilon}(a,b) \le \frac{C\cdot\varepsilon}{|I(b)|}$, and $0 \le R_2(b,a) - R_2^{\varepsilon}(b,a) \le \frac{C\cdot\varepsilon}{|I(b)|}$, which gives $-\frac{C\cdot\varepsilon}{|I(b)|} \le R_2^{\varepsilon}(b,a) - R_2(b,a) \le 0$, and then we get $-\frac{C\cdot\varepsilon}{|I(b)|} \le R_2(a,b) - R_2^{\varepsilon}(a,b) + R_2^{\varepsilon}(b,a) - R_2(b,a) \le \frac{C\cdot\varepsilon}{|I(b)|}$.

By Theorem 1, we get $R_2(a,b) = R_2(b,a)$ by setting $l = 2$. Therefore, $-\frac{C\cdot\varepsilon}{|I(b)|} \le R_2^{\varepsilon}(b,a) - (R_2^{\varepsilon}(a,b) \le \frac{C\cdot\varepsilon}{|I(b)|}$, which gives $|R_2^{\varepsilon}(a,b) - R_2^{\varepsilon}(b,a)| \le \frac{C\cdot\varepsilon}{|I(b)|}$. □

Theorem 4 shows that the similarity of WebSim-pruning is not symmetrical when $\varepsilon > 0$, and gives the maximal difference between $R_2^{\varepsilon}(a,b)$ and $R_2^{\varepsilon}(b,a)$. By setting $\varepsilon = 0$, we can get $R_2^{\varepsilon}(a,b) = R_2^{\varepsilon}(b,a)$, in this case, the similarity of WebSim-pruning is symmetrical.

7. Experimental study

In this section, we report some preliminary experimental results in real networks derived from practical datasets. Experiments were done on a 2.39 GHz Intel(R) Xeon(R) CPU with 64GB main memory, running Windows Server 2008 R2 Enterprise. All algorithms were implemented in C++ and compiled by using Visual Studio 2010.

7.1. Setup

7.1.1. Datasets

We ran our experiments on the following two real datasets:

- **Citation network** We build a citation network by using the high energy physics paper dataset[1] (Gehrke, Ginsparg, & Kleinberg, 2003; Leskovec, Kleinberg, & Faloutsos, 2005), where each node corresponds to a paper and each edge corresponds to a citation between papers. We get 16,000 papers by breadth first traversal from a randomly chosen paper and get 66,794 citations among these papers.
- **E-mail network** We extract an e-mail network from Enron e-mail dataset[2]. In this e-mail network, a node represents an e-mail user and an edge implies the delivery relationship from one user to another. We choose 16,000 e-mail users by breadth first traversal from a randomly chosen e-mail user and get 32,120 delivery relationship among these e-mail users, the sum of all the delivery frequencies is 233,023.

7.1.2. Evaluation and comparison methods

The effectiveness of the returned rankings is evaluated by NDCG (Normalized Discounted Cumulative Gain) (Järvelin & Kekäläinen, 2002). NDCG value at position k (NDCG@k) of the returned rankings is computed by the exact SimRank scores at iteration 10. Formally, NDCG@k is defined as:

$$\mathrm{NDCG@}k = \frac{\mathrm{DCG@}k}{\mathrm{IDCG@}k} \tag{12}$$

where DCG@k (Discounted Cumulative Gain at k) is defined as:

$$\mathrm{DCG@}k = \begin{cases} \mathrm{REL}(v, v_i), & \text{if } i < 2 \\ \mathrm{DCG@}i + \sum_{i=2}^{k} \frac{\mathrm{REL}(v, v_i)}{\log_2 i}, & \text{if } i \ge 2 \end{cases} \tag{13}$$

where i denotes rank of v_i in the returned list, REL(v, v_i) denotes the exact SimRank score between v and v_i. IDCG@k is a normalized factor to ensure the exact ranking generate equals 1.

In terms of the efficiency of similarity search, we report the time cost of similarity computation, query processing and index building, as well as the space cost of similarity matrix and partial sums index.

We compare our approaches, including WebSim and optimized WebSim (Opt-WebSim), with (1) SimRank that computes similarities by the naive SimRank algorithm (Jeh & Widom, 2002); and (2) optimized SimRank (Opt-SimRank) that speeds up the SimRank computation based on the partial sums (Lizorkin et al., 2010). We use SimRank@2 and SimRank@10 to denote the SimRank at iteration 2 and 10 respectively. Similarly, we use Opt-SimRank@2 and Opt-SimRank@10 to denote the Opt-SimRank at iteration 2 and 10 respectively. All the comparison methods are implemented strictly following their papers. The decay factors of WebSim, SimRank and Opt-SimRank are all set as 0.8, since there is little difference in relative rankings on different values (Jeh & Widom, 2002).

7.2. Effectiveness comparison

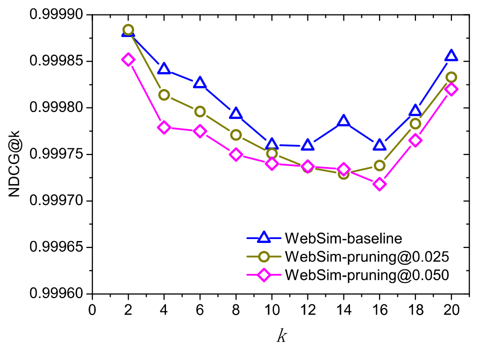
7.2.1. NDCG value

Fig. 2 a shows the NDCG change versus position k on citation network. At each position k, the NDCG of WebSim-baseline is higher than both WebSim-pruning@0.025 (WebSim-pruning at threshold 0.025) and WebSim-pruning@0.050 (WebSim-pruning at threshold 0.050), but the difference of them is very minor, which demonstrates the reasonability of our pruning algorithm. From $k = 2$ to 10, there is a downward trend on the curves, this is because each entity are similar to itself, so the NDCG at $k = 1$ is 1, which affects the NDCG from $k = 2$ to 10. The NDCG at each position k of the SimRank rankings are always 1, which are not repeatedly shown in our experiments. Fig. 2b shows the NDCG change versus position k on e-mail network. Compared to the result on citation network, the curves of this result are not so close, this is because different methods may be suit for different datasets, which is common for similarity search algorithms. On both datasets, WebSim achieves on average 99.98% NDCG, which demonstrates WebSim is effective as well as SimRank.

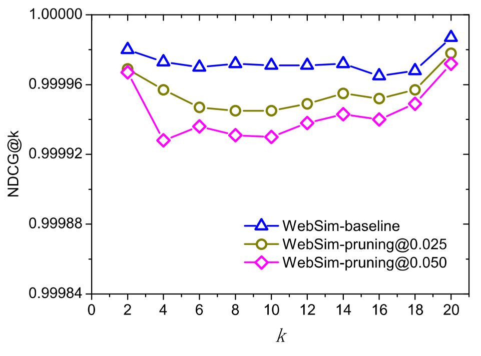
7.2.2. Case study on citation network

Next we give some case studies on citation network by making use of WebSim. The top 10 similar papers for different queries are shown in Table 1. The first column shows the returned list of the query “Adiabatic Motion of a Quantum Particle in a Two-Dimensional Magnetic Field” that is a paper on the adiabatic motion of “Quantum Particle”. We found that most of the returned papers are highly relevant to the given query. For example, “Adiabatic Motion of a Quantum Particle in a Two-Dimensional Magnetic Field” is the query itself; “Embedding of Relativistic Particles and Spinor Natural-Frame” is a paper on “Relativistic Particles” and “Spinor Natural-Frame”, which is relevant to “Quantum Particle” even though there is no common term in their titles; “Fermion Quantum Numbers and Families Replication from an Extension of Space-Time Relativity” introduced the motion of particles on “1+3 space-time”, which is also relevant to the motion of “Quantum Particle”; “Effective Dynamics on a Line” analyzed the effective classical/quantum dynamics of a particle constrained on a closed line embedded in a higher dimensional configuration space, which is relevant to motion of “Quantum Particle” as well; “Dirac Particles in Twisted Tubes” discussed the motion of Dirac particle in the interior of a twisted tube with boundary conditions, “Quantum Charged Spinning Particles in a Strong Magnetic Field (a Quantal Guiding Center Theory)” presented a quantal guiding center theory which allows to systematically study the separation of the different time scale behaviors of a quantum charged spinning particle moving in an external inhomogeneous magnetic field, both of

[1] http://snap.stanford.edu/data/cit-HepPh.html
[2] http://www.cs.cmu.edu/~enron/



(a) Citation network



(b) E-mail network

Fig. 2. NDCG value change on varying *k*.

Table 1
Case studies for different queries on citation network.

Rank	Adiabatic motion of a quantum particle in a two-dimensional magnetic field	Brane gases in the early universe	Nonlinear interaction between electromagnetic fields at high temperature	On tree form-factors in (supersymmetric) Yang–Mills theory	Constraints on Brane-localized gravity
1	Adiabatic motion of a quantum particle in a two-dimensional magnetic field	Brane gases in the early universe	Nonlinear interaction between electromagnetic fields at high temperature	On tree form-factors in (supersymmetric) Yang-Mills theory	Constraints on Brane-localized gravity
2	Embedding of relativistic particles and spinor natural-frame	Hybrid superstrings on singular calabi-Yau fourfolds	Large *N* phase transition in continuum QCD_2	On form-factors in Sin(h)-Gordon theory	Dynamical localization of gravity
3	Fermion quantum numbers and families replication from an extension of space-time relativity	On T-duality in Brane gas cosmology	A simple derivation of the hard thermal loop effective action	New representation for Lagrangians of self-dual nonlinear electrodynamics	Constraints on AdS_5 Embeddings
4	Effective dynamics on a line	Loitering phase in Brane gas cosmology	The Barton expansion and the path integral approach in thermal field theory	An infinite set of tree amplitudes in Higgs–Yang–Mills	A Brane world solution to the cosmological constant problem
5	Dirac particles in twisted tubes	Numerical experiments in string cosmology	Integral representations of thermodynamic 1PI Green functions in the world-line formalism	Gravitational SD Perturbiner	Inflation and gauge hierarchy in Randall–Sundrum compactification
6	Quantum charged spinning particles in a strong magnetic field (a quantal guiding center theory)	Effective tachyonic potential in closed string field theory	Thermal matter and radiation in a gravitational field	Gravitationally dressed Parke–Taylor amplitudes	Non-conventional cosmology from a brane-universe
7	Dimensional reduction by a two-Form (another alternative to compactification)	Acoustics in Bose–Einstein condensates as an example of Lorentz symmetry breaking	Non-linear electromagnetic interactions in thermal QED	SD Perturbiner in YM+Gravity	Dynamic dilatonic domain walls
8	Charged Particles in a 2+1 Curved Background	Relation between Tunneling and Particle Production in Vacuum Decay	Thermal Green's Functions from Quantum Mechanical Path Integrals II: Inclusion of Fermions	On amplitudes in self-dual sector of Yang-Mills theory	Brane cosmological evolution in a bulk with cosmological constant
9	Non-minimal couplings in two dimensional gravity: a quantum investigation	Brane gas cosmology in M-theory: late time behavior	Non-linear electromagnetic interactions in thermal QED	Nonlinear Self-Duality and Supersymmetry	Modeling the fifth dimension with scalars and gravity
10	Quantum Mechanical Embedding of Spinning Particle and Induced Spin-connection	Complete Wetting of Gluons and Gluinos	Thermal Green's Functions from Quantum Mechanical Path Integrals	Electric-Magnetic Duality Rotations in Non-Linear Electrodynamics	An Alternative to Compactification

them are obviously relevant to the given query. Similarly, other papers in the returned list are all similar to the given query, including "Charged Particles in a 2+1 Curved Background", "Non-minimal couplings in two dimensional gravity: a quantum investigation" and "Quantum Mechanical Embedding of Spinning Particle and Induced Spin-connection".

The second column shows the returned list of the query "Brane Gases in the Early Universe", which is a paper on "D-branes" in string theory. In which, the "Brane Gases in the Early Universe" in the returned list is the query itself; "Hybrid Superstrings on Singular Calabi-Yau Fourfolds" discussed the "wound strings" and outlined some possible extensions, including higher-dimensional wrapped branes, which is relevant to "D-branes"; "On T-Duality in Brane Gas Cosmology" is also on "D-branes", which is relevant to the given query as well. Other papers in the returned list from rank 3 to 10 can be analyzed similarly, which are also relevant to the given query.

Similar cases can be found in other queries, such as "Nonlinear interaction between electromagnetic fields at high temperature", "On tree form-factors in (supersymmetric) Yang–Mills theory" and "Constraints on Brane-Localized Gravity", the returned list of these queries

Table 2
Pre-computation time (min).

Dataset	SimRank@2	SimRank@10	Opt-SimRank@2	Opt-SimRank@10	WebSim	Opt-WebSim
Citation	75.66	435.72	43.45	189.38	40.08	0.01
E-mail	33.93	278.56	24.10	124.55	13.59	0.04

Table 3
Similarity matrix size (K).

Dataset	SimRank@2	SimRank@10	Opt-SimRank@2	Opt-SimRank@10	WebSim	Opt-WebSim
Citation	2923.51	28435.66	2923.60	28435.66	362.37	362.37
E-mail	71095.38	113033.86	71095.37	113033.86	2387.90	2387.90

can be analyzed similarly. From above discussions, we can conclude that our proposed WebSim can really reflect the reality to single out similar results.

7.3. Efficiency comparison

7.3.1. Pre-computation cost

Table 2 shows the running time for pre-computing similarity matrices. On both datasets, SimRank@2 requires less time cost than SimRank@10, since it computes only the 2-hop similarities. Similarly, Opt-SimRank@2 also requires less time cost than Opt-SimRank@10. The time cost of WebSim is lower than the comparison methods, since it computes only the 1-hop similarities. On citation network, the time cost of WebSim is 52.96% of SimRank@2, 9.20% of SimRank@10, 92.24% of Opt-SimRank@2 and 21.64% of Opt-SimRank@10 respectively; and on e-mail network is 40.05%, 48.78%, 56.37% and 10.91% respectively. By Opt-WebSim, the efficiency of similarity computation can be further improved. As shown in this table, the time cost of the comparison methods is reduced by 99.97% and 99.70% respectively on the citation and e-mail networks.

Table 3 shows the size of non-zero similarity matrix on the citation and e-mail networks. Here the symbol “K” in the caption refers to “2^{10}”. On both datasets, the similarity matrix of SimRank@2 is smaller than SimRank@10, since SimRank@10 involves a lot of entries of lower similarity scores. Similarly, the similarity matrix of Opt-SimRank@2 is smaller than Opt-SimRank@10. The similarity matrix size of both SimRank@2 and Opt-SimRank@2 are same, since Opt-SimRank reduces only the time cost of SimRank, the matrix for storing similarities is still required, similar cases can be found in other methods. The similarity matrices of both WebSim and Opt-WebSim are smaller than other methods, since they compute only the 1-hop similarities. On citation network, the similarity matrix size of our approach is 12.39% of SimRank@2 and 1.27% of SimRank@10 respectively; and on e-mail network, the similarity matrix size of WebSim is 3.36% of SimRank@2 and 2.11% of SimRank@10 respectively. We find that the improvement of the former is more evidently, this is because the citation network is more sparser than the e-mail network. Compared to both SimRank and Opt-SimRank, the reduction in space cost is 87.60% and 96.64% and respectively on the citation and e-mail networks.

We also recorded the time cost of index building. On citation network, the time cost at $\varepsilon = 0, 0.025, 0.05$ is 2056 ms, 1069 ms and 1019 ms respectively; and on e-mail network is 64,522 ms, 21,954 ms and 16,753 ms respectively. This result demonstrates that the time cost of index building is very low, and shows a downward trend as ε increases.

7.3.2. On-line query processing

In order to accurately test the average execution time of query processing, we randomly choose 5000 query nodes and process each query with 30 runs, and then divide the total time cost by “5000∗30”. The average execution time of query processing on both citation and e-mail networks is shown in Table 4, where $k = 50$ and $\varepsilon = 0.025$. We find that the time cost of SimRank@10 and Opt-SimRank@10 is close since their similarity matrix sizes are same, similar case can be found in SimRank@2 and Opt-SimRank@2. The query time of SimRank@10 is higher than SimRank@2, since SimRank@10 needs to check more entities of lower similarity scores. The result of Opt-SimRank@10 and Opt-SimRank@2 can be explained similarly. On both datasets, WebSim-baseline is the most time-consuming, since it computes the similarity between query and candidate in the on-line stage, which increases the time cost of query processing. The time cost of WebSim-pruning is significantly lower than other methods, since it searches the similar result from only a partial sums index. These results demonstrate that WebSim-pruning is more efficient than other methods.

7.4. On varying threshold ε

We next report some results to test the effect of threshold ε. Fig. 3 illustrates the NDCG, index size, index building time and query processing time versus threshold ε on citation network. From Fig. 3a, we can clearly see that the NDCG decreases as ε increases except $\varepsilon = 0.05$. This is because the partial sums lower than ε are pruned when processing queries, which affects the effectiveness of returned rankings. From Fig. 3b, c and d, we find that the curves show a downward trend as ε increases. This is because the partial sums lower than ε are skipped when building index, which naturally decreases the index size, index building time and query processing time.

Fig. 4 illustrates the NDCG, index size, index building time and query processing time versus threshold ε on e-mail network. We can clearly see that the curves of Fig. 4a, b, c and d also show a downward trend with ε increasing. This phenomenon is similar to the result on citation network and can be explained similarly. Compared to Fig. 3, the downward trend of the curves in Fig. 4 is more evident. This is because there are more index items of lower partial sums in e-mail network, and these items would be pruned with ε increasing.

Table 4
Average execution time of on-line query processing on $k = 50$ (ms).

Dataset	SimRank@2	SimRank@10	Opt-SimRank@2	Opt-SimRank@10	WebSim-baseline	WebSim-pruning@0.025
Citation	0.31	3.29	0.31	3.19	102.29	0.14
E-mail	10.40	16.27	10.10	16.82	56.88	0.56

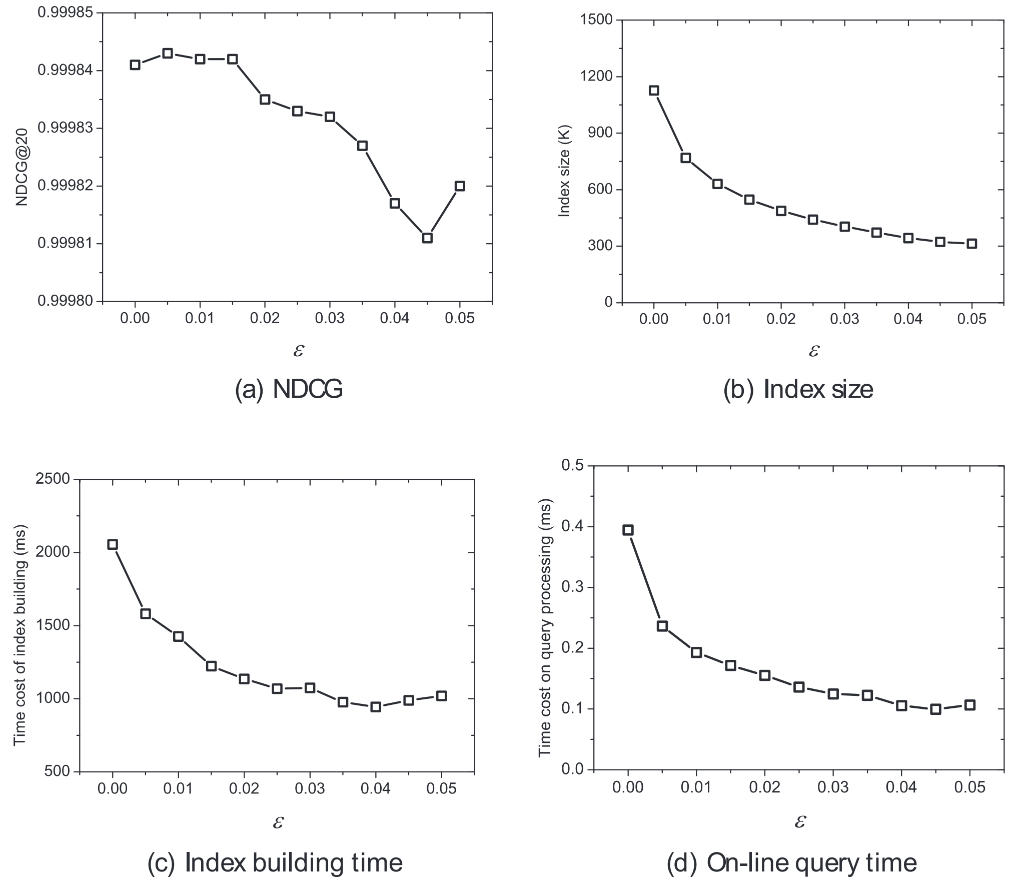


Fig. 3. Test threshold ε on citation network.

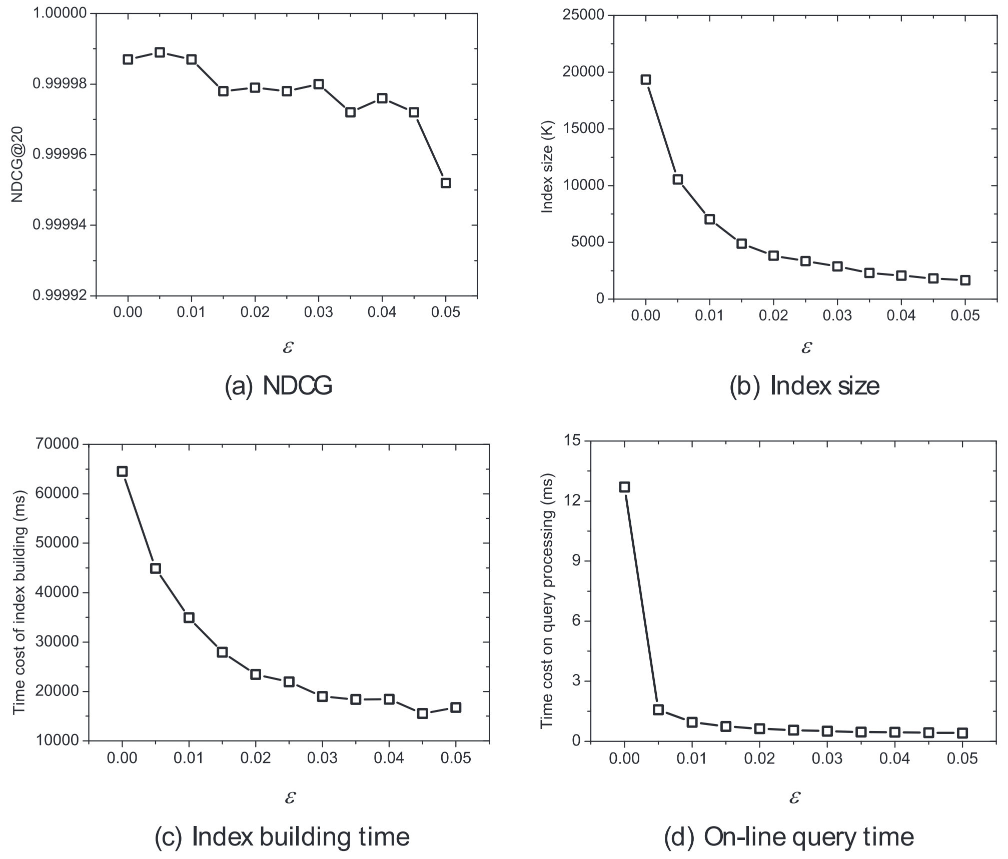


Fig. 4. Test threshold ε on e-mail network.

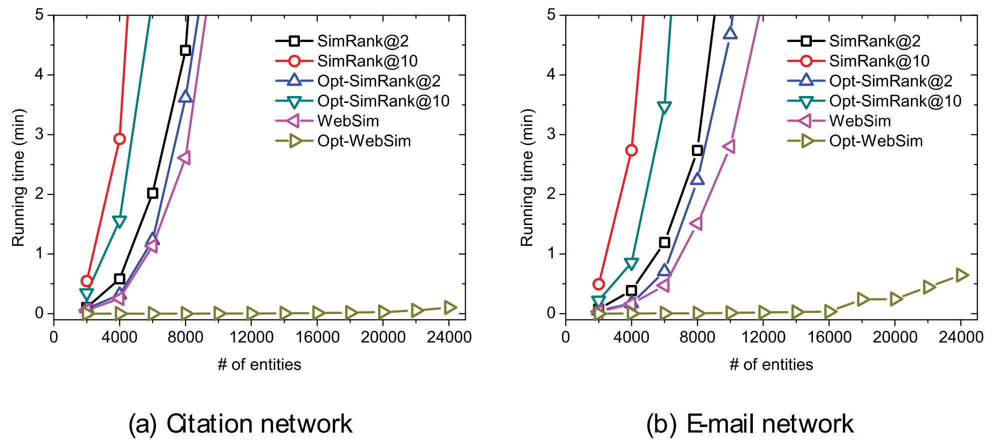


Fig. 5. Running time v.s. # of entities.

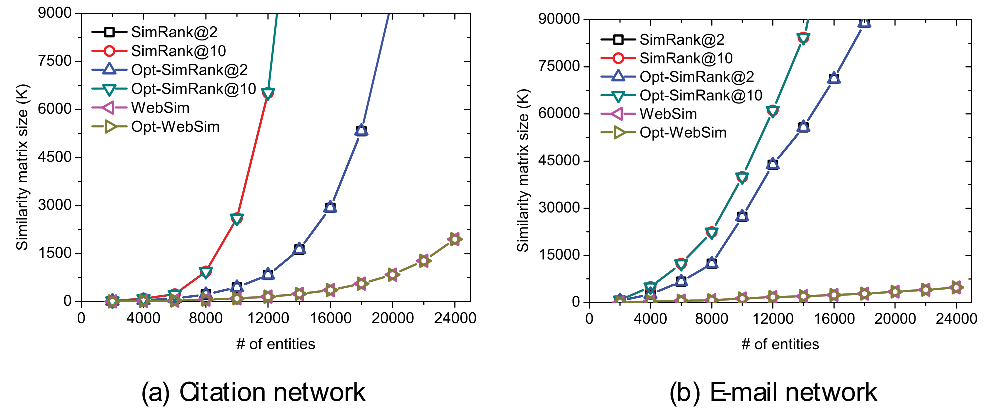


Fig. 6. Similarity matrix size v.s. # of entities.

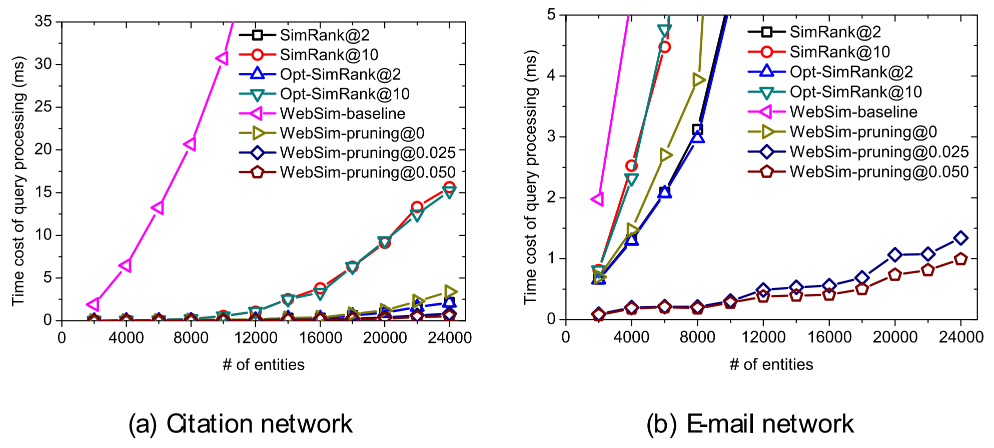


Fig. 7. Time cost of query processing v.s. # of entities.

7.5. Scalability

Next we test the performance of WebSim with increasing entities through comparison with SimRank and Opt-SimRank. Fig. 5 shows the running time for computing similarity matrix on varying entity number. On both datasets, the time cost of WebSim is lower than the comparison methods, but the improvement is not so evident. The time cost of Opt-WebSim is significantly lower than other methods, since the accumulation operations on zero similarity scores are skipped when computing similarities. On varying entity number, the incremental time of Opt-WebSim is always lower than other methods, which shows a good performance in pre-computation stage.

Fig. 6 shows the similarity matrix size on varying entity number. The similarity matrices of both WebSim and Opt-WebSim are significantly smaller than the comparison methods. This result demonstrates that our approach can efficiently reduce the space cost for similarity computation when the network grows large. The curves of WebSim and Opt-WebSim are overlapped since their space complexities are same, other overlapped curves can also be explained similarly.

Fig. 7 shows the execution time of on-line query processing on varying entity number. The time cost of WebSim-baseline is higher than other methods, since the similarity between query and candidate is computed on-line, which increases time cost of query processing. WebSim-pruning performs better than other methods since the searching space can be significantly reduced by removing the items of lower partial sums from the partial sums index. The curves of SimRank@2 and Opt-SimRank@2 are almost overlapped, this is because the sizes of their similarity matrices are same, which gives same searching spaces, and similar cases can be found in other curves. With entity increasing, the incremental time of WebSim-pruning is always minor, which demonstrates that WebSim-pruning can support fast query processing in large networks.

8. Conclusion

This paper introduced a link-based similarity search method WebSim for efficiently searching similar entities from web networks. Compared to iterative SimRank computation, WebSim computes only the 1-hop similarities in the off-line stage, and the actual similarities at the second iteration are computed on-line based on 1-hop similarities. The pre-computation cost is reduced by an optimized algorithm which skips the unnecessary accumulation operations on zero similarity scores, and the on-line query processing is speeded up by an efficient pruning algorithm which searches similar entities from a partial sums index derived from 1-hop similarities. Empirical studies on real datasets through comparison with SimRank and its optimized algorithms show that WebSim has on average a 99.83% reduction in the time cost and a 92.12% reduction in the space cost of similarity computation, and achieves on average 99.98% NDCG.

The contributions presented in this paper are different from the existing methods. Compared to the multi-hop neighbor-based similarity measures (Kumar, Ye, & Doermann, 2014; Nikolic, 2012; Zhang et al., 2015), WebSim computes only the 1-hop similarities in the off-line stage, in which the large matrix for storing multi-hop similarities is not required. In this respect, there are also some measures which compute similarity based on 1-hop neighbors, such as cosine distance-based similarity (Liao & Xu, 2015; Ye, 2015), ontology-based similarity (Sánchez & Batet, 2013; Sohn, Yim, Lee, & Lee, 2014) and categorical-based similarity (dos Santos & Zárate, 2015). Although these methods do not involve the expensive operations for iterative similarity computation, however, the indirect connections would be neglected when computing similarities. Unlike these methods, WebSim utilizes 2-hop neighbors for similarity computation during online query processing, which considers not only direct connections but also indirect connections.

This work has both theoretical and practical implications. Theoretically, WebSim particularly contributes to the computation of existing similarity measures. For example, it can speed up the computation of the similarity measures (Xi et al., 2005; Zhang et al., 2015; Zhao et al., 2009) that are defined with respect to a "random surfer" model by restricting the iteration number of similarity computation. WebSim can also enhance the effectiveness of the 1-hop neighbor-based similarity measures by further integrating 2-hop neighbors into their similarity computations. Besides, WebSim can be regarded as a general method, since it can be easily combined with existing optimization techniques (Du et al., 2015; Yu et al., 2014; Yu et al., 2015). As far as practical implications are concerned, WebSim can be applied to many web applications, such recommender systems (Champiri, Shahamiri, & Salim, 2015), link prediction (He, Liu, Hu, & Wang, 2015) and data quality assessment (Mendi, 2015). WebSim provides an effective and efficient solution to evaluate underlying similarity, which helps improve the system performance and map human intuition under different real settings of web data.

The advantages of this paper can be summarized as follows. First, WebSim requires less time and space cost than the multi-hop neighbor-based similarity measures, since the similarity matrix of 1-hop is significantly smaller than that of multi-hop, in which the expensive operations for iterative SimRank computation are saved. Second, WebSim is more effective than the 1-hop neighbor-based similarity measures, since the indirect connections are integrated into similarity computation, which helps find more comprehensive results. Third, the pruning algorithm for on-line query processing is more efficient than the methods (Milchevski, Anand, & Michel, 2015; Zhang et al., 2015) which compute similarities in the on-line stage, since the searching space can be reduced by skipping the index items that are lower than a given threshold.

There are still some limitations in our research. First, unlike the similarity measures (Sun et al., 2011; Zhang et al., 2015) that are on heterogeneous networks, WebSim is proposed for homogeneous networks, in which the importance of different type relationships is not addressed. Second, different from the incremental SimRank algorithms (Du et al., 2015; Li et al., 2010a; Yu et al., 2014), WebSim focuses on only the networks that are static. When network changes, WebSim needs to compute the whole similarity matrix even only a small portion of them is required, which wastes a lot of time and space. Finally, compared to the semantic similarity measures (Duong, Nguyen, Truong, & Nguyen, 2015; He & Tan, 2015), WebSim cannot support similarity search in semantic networks, since the semantics of relationships are not considered for similarity computation.

In future work, we will focus on the following aspects to overcome the above limitations. First, extending our approach for heterogeneous network can be regarded as a promising direction of future work. To this end, we plan to develop a general framework for unifying the different heterogeneous relationships. Second, we intend to develop a new semantic similarity measure for handling the semantic networks by integrating semantic similarity into WebSim. Finally, we would like to extend WebSim for similarity computation in dynamic networks. For this purpose, we will develop an increment algorithm to update the affected areas of the similarity matrix without computing the whole similarities.

Acknowledgment

This work was supported by Innovation Program of Shanghai Municipal Education Commission Grants 15ZZ074, 13YZ075 and 13ZZ111; Bidding Program of Shanghai Research Institute of Publishing and Media Grants SAYB1410; Young University Teacher Funding Program of Shanghai Municipal Education Commission Grants ZZSLG14021; National Science Foundation of China Grants 61170007 and 61202376; and National High-tech R&D Program of China (863 Program) Grants SS2015AA011809.

References

Akmal, S., Shih, L., & Batres, R. (2014). Ontology-based similarity for product information retrieval. *Computers in Industry, 65*(1), 91–107. doi:10.1016/j.compind.2013.07.011.
Amsler, R. (1972). *Application of citation-based automatic classification*. The University of Texas at Austin Linguistics Research Center: Technical report.
Champiri, Z. D., Shahamiri, S. R., & Salim, S. S. B. (2015). A systematic review of scholar context-aware recommender systems. *Expert Systems with Applications, 42*(3), 1743–1758. doi:10.1016/j.eswa.2014.09.017.
Du, L., Li, C., Chen, H., Tan, L., & Zhang, Y. (2015). Probabilistic simrank computation over uncertain graphs. *Information Science, 295*, 521–535. doi:10.1016/j.ins.2014.10.030.
Duong, T. H., Nguyen, N. T., Truong, H. B., & Nguyen, V. H. (2015). A collaborative algorithm for semantic video annotation using a consensus-based social network analysis. *Expert Systems With Applications, 42*(1), 246–258. doi:10.1016/j.eswa.2014.07.046.
Faloutsos, M., Faloutsos, P., & Faloutsos, C. (1999). On power-law relationships of the internet topology. In *Proceedings of the Sigcomm* (pp. 251–262).
Ganesan, P., Garcia-Molina, H., & Widom, J. (2003). Exploiting hierarchical domain structure to compute similarity. *ACM Transactions on Information Systems, 21*(1), 64–93.
Gehrke, J., Ginsparg, P., & Kleinberg, J. M. (2003). Overview of the 2003 kdd cup. *SIGKDD Explorations, 5*(2), 149–151.
He, J., Liu, H., Yu, J. X., Li, P., He, W., & Du, X. (2014). Assessing single-pair similarity over graphs by aggregating first-meeting probabilities. *Information Systems, 42*, 107–122. doi:10.1016/j.is.2013.12.008.
He, Y., Liu, J. N., Hu, Y., & Wang, X. (2015). OWA operator based link prediction ensemble for social network. *Expert Systems With Applications, 42*(1), 21–50. doi:10.1016/j.eswa.2014.07.018.
He, Y., & Tan, J. (2015). Study on SINA micro-blog personalized recommendation based on semantic network. *Expert Systems With Applications, 42*(10), 4797–4804. doi:10.1016/j.eswa.2015.01.045.

Järvelin, K., & Kekäläinen, J. (2002). Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems, 20*(4).

Jeh, G., & Widom, J. (2002). Simrank: a measure of structural-context similarity. In *Proceedings of the Kdd* (pp. 538–543).

Kessler, M. M. (1963). Bibliographic coupling between scientific papers. *American Documentation, 14*, 10–25.

Kumar, J., Ye, P., & Doermann, D. S. (2014). Structural similarity for document image classification and retrieval. *Pattern Recognition Letters, 43*, 119–126. doi:10.1016/j.patrec.2013.10.030.

Lee, P., Lakshmanan, L. V. S., & Yu, J. X. (2012). On top-k structural similarity search. In *Proceedings of the Icde* (pp. 774–785).

Leskovec, J., Kleinberg, J. M., & Faloutsos, C. (2005). Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proceedings of the Kdd* (pp. 177–187).

Li, C., Han, J., He, G., Jin, X., Sun, Y., Yu, Y., et al. (2010a). Fast computation of simrank for static and dynamic information networks. In *Proceedings of the Edbt* (pp. 465–476).

Li, L., Li, C., Chen, H., & Du, X. (2013). Mapreduce-based simrank computation and its application in social recommender system. In *Proceedings of the IEEE international congress on big data, bigdata congress 2013, june 27 2013-july 2, 2013* (pp. 133–140). doi:10.1109/BigData.Congress.2013.26.

Li, P., Liu, H., Yu, J. X., He, J., & Du, X. (2010b). Fast single-pair simrank computation. In *Proceedings of the Sdm* (pp. 571–582).

Liao, H., & Xu, Z. (2015). Approaches to manage hesitant fuzzy linguistic information based on the cosine distance and similarity measures for hfltss and their application in qualitative decision making. *Expert Systems with Applications, 42*(12), 5328–5336. doi:10.1016/j.eswa.2015.02.017.

Lizorkin, D., Velikhov, P., Grinev, M. N., & Turdakov, D. (2010). Accuracy estimate and optimization techniques for simrank computation. *VLDB Journal, 19*(1), 45–66.

Maguitman, A. G., Menczer, F., Erdinc, F., Roinestad, H., & Vespignani, A. (2006). Algorithmic computation and approximation of semantic similarity. *World Wide Web, 9*(4), 431–456.

Mendi, E. (2015). Image quality assessment metrics combining structural similarity and image fidelity with visual attention. *Journal of Intelligent and Fuzzy Systems, 28*(3), 1039–1046. doi:10.3233/IFS-141387.

Milchevski, E., Anand, A., & Michel, S. (2015). The sweet spot between inverted indices and metric-space indexing for top-k-list similarity search. In *Proceedings of the 18th international conference on extending database technology, EDBT 2015, brussels, belgium, march 23-27, 2015.* (pp. 253–264). doi:10.5441/002/edbt.2015.23.

Nikolic, M. (2012). Measuring similarity of graph nodes by neighbor matching. *Intelligent Data Analysis, 16*(6), 865–878. doi:10.3233/IDA-2012-00556.

Sánchez, D., & Batet, M. (2013). A semantic similarity method based on information content exploiting multiple ontologies. *Expert Systems with Applications, 40*(4), 1393–1399. doi:10.1016/j.eswa.2012.08.049.

dos Santos, T. R. L., & Zárate, L. E. (2015). Categorical data clustering: What similarity measure to recommend? *Expert Systems with Applications, 42*(3), 1247–1260. doi:10.1016/j.eswa.2014.09.012.

Small, H. G. (1973). Co-citation in the scientific literature: a new measure of the relationship between two documents. *Journal of the American Society for Information Scienc, 24*(4), 265–269.

Sohn, M. M., Yim, J. H., Lee, S., & Lee, H. J. (2014). Ontology-based dynamic and semantic similarity calculation method for case-based reasoning. *Intelligent Automation & Soft Computing, 20*(1), 33–46. doi:10.1080/10798587.2013.873303.

Sun, Y., Han, J., Yan, X., Yu, P. S., & Wu, T. (2011). *Pathsim: meta path-based top-k similarity search in heterogeneous information networks: Vol. 4* (pp. 992–1003).

Tao, W., Yu, M., & Li, G. (2014). Efficient top-k simrank-based similarity join. *PVLDB, 8*(3), 317–328.

Xi, W., Fox, E. A., Fan, W., Zhang, B., Chen, Z., Yan, J., et al. (2005). Simfusion: measuring similarity using unified relationship matrix. In *Proceedigns of the Sigir* (pp. 130–137).

Xu, J., Li, C., Chen, H., & Sun, H. (2015). Simrank based top-k query aggregation for multi-relational networks. In *Proceedings of the Web-age information management – 16th international conference, WAIM 2015, Qingdao, China, June 8–10, 2015.* (pp. 544–548). doi:10.1007/978-3-319-21042-1_59.

Ye, J. (2015). Improved cosine similarity measures of simplified neutrosophic sets for medical diagnoses. *Artificial Intelligence in Medicine, 63*(3), 171–179. doi:10.1016/j.artmed.2014.12.007.

Yin, X., Han, J., & Yu, P. S. (2006). Linkclus: Efficient clustering via heterogeneous semantic links. In *Proceedings of the Vldb* (pp. 427–438).

Yu, W., Lin, X., & Zhang, W. (2014). Fast incremental simrank on link-evolving graphs. In *Proceedings of the IEEE 30th international conference on data engineering, chicago, ICDE 2014, il, usa, march 31 - april 4, 2014* (pp. 304–315). doi:10.1109/ICDE.2014.6816660.

Yu, W., Lin, X., Zhang, W., Chang, L., & Pei, J. (2013). More is simpler: effectively and efficiently assessing node-pair similarities based on hyperlinks. *PVLDB, 7*(1), 13–24.

Yu, W., Lin, X., Zhang, W., & McCann, J. A. (2015). Fast all-pairs simrank assessment on large graphs and bipartite domains. *IEEE Transactions on Knowledge and Data Engineering, 27*(7), 1810–1823. doi:10.1109/TKDE.2014.2339828.

Zhang, M., He, Z., Hu, H., & Wang, W. (2012). E-rank: a structural-based similarity measure in social networks. In *Proceedings of the 2012 IEEE/WIC/ACM international conferences on web intelligence, WI 2012, macau, china, december 4–7, 2012* (pp. 415–422). doi:10.1109/WI-IAT.2012.111.

Zhang, M., Hu, H., He, Z., & Wang, W. (2015). Top-k similarity search in heterogeneous information networks with x-star network schema. *Expert Systems With Applications, 42*(2), 699–712. doi:10.1016/j.eswa.2014.08.039.

Zhao, P., Han, J., & Sun, Y. (2009). P-rank: a comprehensive structural similarity measure over information networks. In *Proceedings of the Cikm* (pp. 553–562).

Zhao, X., Xiao, C., Lin, X., Liu, Q., & Zhang, W. (2013). A partition-based approach to structure similarity search. *PVLDB, 7*(3), 169–180.