Information Sciences 381 (2017) 1–19

Contents lists available at ScienceDirect

Information Sciences

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

Fast top-k similarity join for SimRank

Ruiqi Li[a], Xiang Zhao[a,c,*], Haichuan Shang[b], Yifan Chen[a], Weidong Xiao[a,c]

[a] College of Information System and Management, National University of Defense Technology, Changsha, Hunan, 410072, China
[b] National Institute of Information and Communication Technology, Koganei, Tokyo, 184-8795, Japan
[c] Collaborative Innovation Center of Geospatial Technology, Wuhan, Hubei, 410000, China

ARTICLE INFO

Article history:
Received 10 March 2016
Revised 6 September 2016
Accepted 22 October 2016
Available online 15 November 2016

Keywords:
Top-k
Similarity join
Iterative pruning
Upper bound

ABSTRACT

SimRank is a well-studied similarity measure between two nodes in a network. However, evaluating SimRank of all nodes in a network is not only time-consuming but also not pragmatic, since users are only interested in the most similar pairs in many real-world applications. This paper focuses on top-k similarity join based on SimRank. In this work, we first present an incremental algorithm for computing SimRank. On top of that, we derive an iterative batch pruning framework, which is able to iteratively filter out unpromising nodes and obtain the top-k pairs in a fast mode. Specifically, we define the concept of super node such that for a node in the network, the SimRank with its super node is not less than that with any others. Based on this feature, we propose a tight upper bound for each node that can be easily calculated after each iteration. Experiments on both real-life and synthetic datasets demonstrate that our method achieves better performance and scalability, in comparison with the state-of-the-art solution.

1. Introduction

SimRank is a popular link-based similarity measure to evaluate node pair similarities in a network. It is widely adopted in many real-world applications including sponsored search, web spam detection, schema matching, etc. The basic intuition behind SimRank is based on the object-to-object relationship found in many domains of interest; that is, two objects are similar if they are referenced by similar objects. By recursively computing the similarity between two nodes based on the similarities between their neighbors, SimRank measures the similarity of structural context, which can be applied to any domain where there are enough relevant relationships among objects. For example, in a social network, SimRank is a useful function to identify similar users and suggest potential friends to users, which can further help predict possible links and trace information disseminated over the network. As to computational challenges, significant efforts have been devoted to efficient SimRank computation. Consider a (directed) network G with n nodes. The original iterative algorithm [4] computes all-pair SimRank in $O(\xi d^2 n^2)$ time in ξ iterations, where d is the average in-degree of the network. It is improved to $O(\xi dn^2)$ time via partial sum memorization [9], and further expedites through fine-grained memorization [17]. Albeit, computing all-pair SimRank in real-life large networks can be still prohibitive in terms of both time and space costs, seeing the ever-growing sizes of networks. As an alternative, identifying node pairs of highest SimRank meets the needs of retrieving only the highly similar node pairs. Most of the existing methods require a similarity threshold from users [6,18], and

* Corresponding author at: College of Information System and Management, National University of Defense Technology, Changsha, Hunan Province, 410072, China.
E-mail address: xiangzhao@nudt.edu.cn (X. Zhao).

http://dx.doi.org/10.1016/j.ins.2016.10.042
0020-0255/© 2016 Elsevier Inc. All rights reserved.

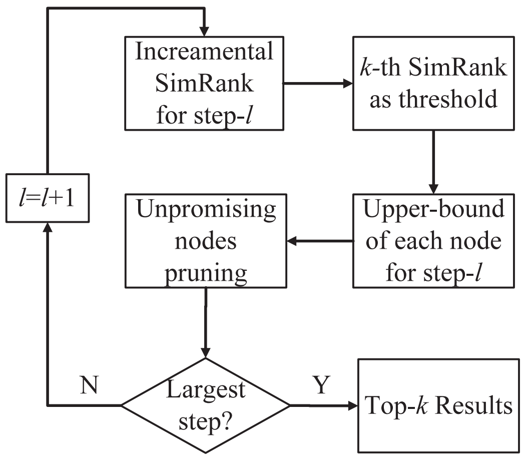
Increamental SimRank for step-l
k-th SimRank as threshold
Upper-bound of each node for step-l
Unpromising nodes pruning
l=l+1
Largest step?
N
Y
Top-k Results

Fig. 1. Overview of KSimJoin.

only node pairs whose similarities exceed the threshold are returned. Unluckily, it can be formidable to select an appropriate threshold that guarantees decent results yet without irrelevant pairs. Lately, an appealing approach is introduced to find k most similar node pairs by SRK-Join [14], without the requirement of a threshold as input. In particular, it defines *second meeting probability* (cf. Section 3), related to each node v, to denote the probability of two random paths starting both from node v and firstly meet at another same node again. Leveraging the probabilities, it then encodes each v as a multi-dimensional vector such that SimRank of two nodes can be computed by the *dot product* of the two vectors. To retrieve the top-k pairs, SRK-Join executes two times of scanning of every path that starts from the object nodes. The first is to help to compute a candidate set containing 2k distinct nodes, and the second is to obtain the top-k pairs. SRK-Join is tested to be efficient on real-life networks; nonetheless, we observe margins for further improvement. In particular, (1) in order to encode every node as a vector, SRK-Join requires calculating second meeting probability of every node. Through experiments, however, we observe that calculating second meeting probability for nodes with high in-degrees can be fairly time-consuming, as calculating second meeting probability of a node requires $O(d^{\xi})$ time. Thus, evaluating the first-meeting probability of two paths, regardless of (same or not) starting nodes, via second meeting probability, incurs significant computational overhead. (2) In SRK-Join, the 2k candidate nodes are re-scanned for final results. This results in computational redundancy, since the candidate nodes have already been scanned once. Moreover, the second scanning is dependent on the first scanning, which means the information calculated in the first round can be somehow reused. Thus, we exploit the opportunities for better performance.

In this work, we first present an incremental algorithm of SimRank by iterations, and then encapsulate the procedure into a novel iterative batch pruning framework KSimJoin. We design an upper bound for each node that is both tight and easy to be derived. Specifically, as the iteration goes deeper, the upper bound becomes tighter based on second meeting probabilities newly calculated in the current iteration. Thus, a quantity of nodes will be removed in each iteration before SimRank is eventually calculated. Moreover, computation sharing of the required second meeting probabilities among the iterations is enabled for speedup. Fig. 1 gives an overview of our framework KSimJoin. In each iteration, we incrementally calculate SimRank for candidate nodes and their upper-bounds. The candidate nodes are the whole network in the first iteration. We select the top-k SimRank node pairs and compare the upper-bound of each node with the kth largest SimRank (threshold). Only those with upper-bound larger than the threshold can remain to the next iteration. As soon as the largest step is reached, the top-k SimRank node pairs will be returned as the answer. As for large networks, we put forward further improvement to reduce the running time and memory footprints of the algorithm.

Contribution. To summarize, we make the following contributions.

- We develop an incremental algorithm for partial SimRank, and on top of that, propose an iterative batch pruning framework for top-k similarity joins, which can efficiently discover k node pairs with the largest SimRank in the network.
- We define the concept of super node $\mathcal{V}_s$ of node v such that SimRank between v and its super node is not less than that between v and any other nodes. Based on this, we put forward a tight upper bound to prune unpromising nodes under the framework.
- The resultant algorithm KSimJoin is evaluated on real-life and synthetic networks. Empirical results witness the effectiveness and efficiency of the proposed techniques, suggesting a better option for potential applications.

Organization. Section 2 introduces the basic concepts related to our work, and formally states the problem. Then, we introduce the method for computing partial SimRank in Section 3. We present the iterative batch pruning framework in Section 4, including the design of upper bound. Further improvement for large networks is presented in Section 5. Empirical results are reported and analyzed in Section 6. We discuss related work in Section 7, and conclude the paper in Section 8.