Link-based Analysis on Large Graphs

Presented by
Weiren Yu
Mar 01, 2011
Overview

1. Introduction
2. Problem Definition
3. Optimization Techniques
4. Experimental Results
1. Introduction

- Many applications require a measure of “similarity” between objects.

- Recommender System (amazon.com)
- Citation of Scientific Papers (citeseer.com)
- Web Search Engine (google.com)
- Graph Clustering
Existing Similarity Measures

- **Textual-Content Similarity** (text-based)
  - Vector-cosine similarity, Pearson correlation in IR, ...

- **Structural-Context Similarity** (link-based)
  - **PageRank**: A page’s authority is decided by its neighbors’ authorities.
  - **HITS**: 
    - “a good hub” --- a page that pointed to many other pages
    - “a good authority” --- a page that was linked by many hubs
  - **SimRank**: similar objects are referenced by similar objects.
  - **LinkFusion**: reinforcement assumption
  - **Penetrating-Rank**: entities are similar if
    - (1) they are referenced by similar entities;
    - (2) they reference similar entities.
What is SimRank?

- The similarity in a domain can be modeled as graphs.
  \[ \text{vertices} \rightarrow \text{objects}, \text{edges} \rightarrow \text{relationships} \]

- SimRank is an important similarity measure which exploits the relationships between vertices on web graphs.
  (Glen Jeh & Jennifer Widom, 2002)

- Basic intuition:
  - Two objects are similar if their neighbors are similar.
    (the recursive definition)
  - Objects are maximally similar to themselves.
    (the base case)
Overview

1. Introduction
2. Problem Definition
3. Optimization Techniques
4. Experimental Results
Definition 1 (SimRank similarity)

Let $s : V^2 \rightarrow [0, 1]$ be a similarity function on $G^2$

- if $a = b$, $\Rightarrow s(a, b) = 1$,
- if $I(a)$ or $I(b) = \emptyset$, $\Rightarrow s(a, b) = 0$,
- otherwise:
  
  $$s(a, b) = \frac{C}{|I(a)| \cdot |I(b)|} \cdot \sum_{i=1}^{\left|I(a)\right|} \sum_{j=1}^{\left|I(b)\right|} s(I_i(a), I_j(b))$$

- $C$ is a decay factor btw. 0 & 1
- symmetric: $s(a, b) = s(b, a)$

Similarity btw. $a$ & $b$ is the average similarity btw. in-neighbors of $a$ and in-neighbors of $b$. 
Naïve SimRank Computation

- **Iterative Paradigm:**

  \[ S_0(a, b) = \begin{cases} 
    0 & a \neq b \\
    1 & a = b 
  \end{cases} \]

  \[ S_{k+1}(a, b) = \frac{C}{|I(a)| \cdot |I(b)|} \cdot \sum_{i=1}^{I(a)} \sum_{j=1}^{I(b)} S_k(I_i(a), I_j(b)), \quad k = 0, 1, \ldots \]

- (monotonicity) \( S_k(a, b) \uparrow S(a, b) \) as \( k \to \infty \).
- (symmetry) \( S(a, b) = S(b, a) \).
- (stability) \( S(a, b) \) is independent of \( S_0(a, b) \).
- (complexity) Time: \( O(Kn^2d^2) \), Space: \( O(n^2) \),

  where \( d \) is the average of \(|I(\cdot)|\) over all nodes.
Existing Techniques for SimRank Optimization

- **Deterministic Method** [PVLDB 08]
  - (to compute $s(\cdot, \cdot)$ iteratively for finding a fixed point)
  - \[ s_{k+1}(a,b) = \frac{C}{|I(a)| \cdot |I(b)|} \cdot \sum_{i=1}^{I(a)} \sum_{j=1}^{I(b)} s_k(I_i(a), I_j(b)), \quad k = 0, 1, \ldots \]
  - Advantage: accurate
  - Disadvantage: high time complexity $O(Kn^3)$

- **Probabilistic Method** [WWW 05]
  - (to estimate $s(\cdot, \cdot)$ stochastically by using Monte-Carlo)
  - \[ s(a,b) = E(c^{T(a,b)}) , \text{ where} \]
  - $T(a,b):$ the first meeting time btw. $a$ & $b$
  - Advantage: scalable (linear time)
  - Disadvantage: low similarity quality
Motivation

- The computational time, which has been reduced to $O(Kn^3)$ by [PVLDB08], is still rather costly for practical purposes.

- Optimization for SimRank storage space has not been addressed in scientific literature yet.

- The accuracy estimate $\epsilon = c^k$ in [PVLDB08] is solely based on the empirical inductive method and, therefore, is not preferable.
Our Contributions

- A matrix representation and a storage scheme for SimRank model has been introduced.
  - to reduce space from $O(n^2)$ to $O(m + n)$
  - to improve time from $O(n^3)$ to $O(\min\{n \cdot m, n^r\})$
    in the worst case, where $r \leq \log_2 7$

- Optimization techniques for minimizing the matrix bandwidth have been developed.
  - to improve I/O efficiency

- A successive over-relaxation method has been showed.
  - to accelerate the rate of convergence
3.1 Matrix Representations for SimRank Model

- Let $S = (s_{i,j}) \in \mathbb{R}^{n \times n}$ be a SimRank matrix, where $s_{i,j}$ = the SimRank value btw. vertices $i$ and $j$.
- $P = (p_{i,j}) \in \mathbb{N}^{n \times n}$ be an adjacency matrix, where $p_{i,j}$ = # of edges from vertices $i$ to $j$.

SimRank in matrix notation $O(n^3)$ for matrix multiplication

\[
\begin{cases}
S^{(0)} = I_n \\
S^{(k+1)} = c \cdot Q \cdot S^{(k)} \cdot Q^T \quad \forall I_n \quad (k = 0,1,\ldots)
\end{cases}
\]

- $s_{k+1}(a,b) = \frac{C}{|I(a)| \cdot |I(b)|} \cdot \sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(b)|} s_k(I_i(a), I_j(b))$
- $s_{k+1}(a,b) = c \cdot \sum_{i=1}^n \sum_{j=1}^n \left( \frac{p_{i,a}}{\sum_i p_{i,a}} \right) \cdot s_i^{(k)} \cdot \left( \frac{p_{j,b}}{\sum_j p_{j,b}} \right), \quad k = 0,1,\ldots$
- $S^{(k+1)} = c \cdot Q \cdot S^{(k)} \cdot Q^T \quad (a \neq b)$
3.1 Matrix Representations for SimRank Model (cont.)

- **For dense graphs**
  - Fast matrix multiplication algorithms can be applied to speed up the SimRank computation.
    - Strassen Algorithm: $O(n^r)$, where $r = \log_2 7$
    - Coppersmith-Winograd Algorithm: $O(n^{2.38})$

- **For sparse graphs**
  - Compressed Sparse Row (CSR) are used to represent $Q$ due to its high compression ratio.
  - CSR has $O(m \cdot n)$ time with the space $O(m+n)$. 
3.2 Permuted SimRank Iterative Approach

- The permutation method allows improving I/O efficiency for SimRank computation.
- The main idea involves 2 steps:
  - **Reversed Cuthill-McKee (RCM) algorithm** for non-symmetric matrix is introduced for finding an optimal permutation while reordering the matrix Q during the precomputation phase.
  - **Permuted SimRank iterative equation** is developed for reducing the matrix bandwidth for SimRank computation.
3.2 Permuted SimRank Iterative Approach (cont.)

- The permutation $\pi$ can be thought of as a bijection between the vertices of the labeled graph $G_Q$ and $G_{\pi(Q)}$.
- $\beta(\pi(Q)) \leq \beta(Q)$.
- We extend the original RCM to the directed graph by adding “the mate $Q^T$” and apply RCM to $Q + Q^T$. 

$$\pi = \begin{pmatrix}
1 & 2 & 3 & 4 & 5 & 6 \\
4 & 2 & 1 & 3 & 6 & 5
\end{pmatrix} \quad \text{node } v_i \quad \pi(v_i)$$
3.2 Permuted SimRank Iterative Approach (cont.)

- **Permuted SimRank Equation**
  - Let $\pi$ be an arbitrary permutation with an induced permutation matrix $\Theta$. For a given graph $G$, SimRank similarity score can be computed as

$$S^{(k)} = \pi^{-1}(\hat{S}^{(k)})$$

where

$$\begin{cases} 
\hat{S}^{(0)} = I_n \\
\hat{S}^{(k+1)} = c \cdot \pi(Q) \cdot \hat{S}^{(k)} \cdot \pi(Q)^T \lor I_n, & k = 0, 1, \ldots 
\end{cases}$$

- For the computation to be I/O efficient, $Q$ needs to be preordered during the precomputation phrase.
3.3 Successive Over-relaxation (SOR) SimRank Algorithm

- SOR can be used for computing $S^{(k)}$ to effectively exhibit faster rate of convergence.

- SOR SimRank Equation:
  - Let $Q = (q_{i,j}) \in \mathbb{R}^{n \times n}$, $S^{(k)} = (s_1^{(k)} \ s_2^{(k)} \ \ldots \ \ s_n^{(k)})$, where $s_j^{(k)}$ is the $j$-th column vector of $S^{(k)}$, then

\[
\begin{align*}
    s_i^{GS(k+1)} &= c \cdot Q \cdot \left( \sum_{j<i} q_{i,j} \cdot s_j^{(k)} + \sum_{j>i} q_{i,j} \cdot s_j^{(k+1)} \right) \lor I_n \\
    s_i^{SOR(k+1)} &= (1 - \omega) \cdot s_i^{SOR(k)} + \omega \cdot s_i^{GS(k+1)}
\end{align*}
\]
Overview

1. Introduction
2. Problem Definition
3. Optimization Techniques
4. Experimental Results
4 Experimental Evaluation

Experimental Setup

Hardware
- 2.0GHz Pentium(R) Dual-Core / 2GB RAM
- Windows Vista OS / Visual C++ 6.0

Data Sets
- Synthetic
  - graph with an average of 8 links per page.
  - 10 sample adjacency matrices from 1K to 10K with $\xi \sim \text{uniform}[0; 16]$ out-links on each row.
- Real-life
  - Wikipedia (3.2M articles with 110M intra-wiki links Oct. '07)
  - We choose the relationship: “a category contains an article to be a link from the category to the article“.

Parameter Settings
- $c = 0.8$, $\omega = 1.3$, $\epsilon = 0.05$
Experimental Results

(a) Time Efficiency on Sparse Graphs
(b) Time Efficiency on Dense Graphs
(c) Convergence Rate \((n = 10K, \omega = 1.3)\)
(d) Relaxation Factor \( (n = 10K, \varepsilon = 0.05) \)
(e) I/O Efficiency on Sparse Graphs
(f) Extended RCM \( (m = 5, 496, 208) \)
Thank You!
Conclusions

- We formalized the SimRank equation in matrix notations.
- We investigated optimization issues for SimRank computation.
  - A compressed storage scheme for sparse graphs is adopted for reducing the space from $O(n^2)$ to $O(n + m)$.
  - A fast matrix multiplication for dense graphs is used for improving the time from $O(n^2 \cdot d)$ to $O(\min \{n \cdot m, n^r\})$, $r \leq \log_2 7$.
  - A permuted SimRank iteration was developed in combination of the extended RCM algorithm to achieve its I/O efficiency.
  - A SOR method has been showed to significantly speed up the convergence rate of the SimRank iteration.
- Our experimental evaluations on synthetic and real-life data sets demonstrate the efficiency of our methods.