

A Recommender System Based on Local Random Walks and Spectral Methods

Zeinab Abbassi
Department of Computer Science, UBC
201 2366 Main Mall
Vancouver, Canada
zeinab@cs.ubc.ca

Vahab S. Mirrokni
Microsoft Research
One Microsoft Way
Redmond, WA
mirrokni@microsoft.com

ABSTRACT

In this paper, we design recommender systems for weblogs based on the link structure among them. We propose algorithms based on refined random walks and spectral methods. First, we observe the use of the personalized page rank vector to capture the relevance among nodes in a social network. We apply the local partitioning algorithms based on refined random walks to approximate the personalized page rank vector, and extend these ideas from undirected graphs to directed graphs. Moreover, inspired by ideas from spectral clustering, we design a similarity metric among nodes of a social network using the eigenvalues and eigenvectors of a normalized adjacency matrix of the social network graph. In order to evaluate these algorithms, we crawled a set of weblogs and construct a weblog graph. We expect that these algorithms based on the link structure perform very well for weblogs, since the average degree of nodes in the weblog graph is large. Finally, we compare the performance of our algorithms on this data set. In particular, the acceptable performance of our algorithms on this data set justifies the use of a link-based recommender system for social networks with large average degree.

Categories and Subject Descriptors

H.4 [Information Systems Applications]: Miscellaneous

General Terms

Algorithms, Theory

Keywords

Social Network Analysis, Random Walks, Spectral Methods, Personalized PageRank, Recommender Systems, Weblogs.

1. INTRODUCTION

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Joint 9th WEBKDD and 1st SNA-KDD Workshop '07 (WebKDD/SNA-KDD'07), August 12, 2007, San Jose, California, USA.

Copyright 2007 ACM 978-1-59593-848-0/XX/XX ...\$5.00.

Recommender systems use the opinions of a community of users to help individuals in that community more effectively identify content of interest from a gigantic set of choices [3]. In social networks, recommender systems can be defined using the link structure among nodes of the network. This method is particularly useful for social networks with large average degree. One of the main social networks with high average degree is the network of weblogs.

The rapid development of blogging is one of the most interesting phenomena in the evolution of the web over the last few years. As of December 2006, the web contains over 37 million weblogs, with roughly 40,000 new weblogs and 750,000 new posts created daily[1]. These weblogs span a wide range, from personal journals read mainly by friends, to very popular weblogs visited by millions of readers. All together, these weblogs form a distinct, human-generated subset of the web (blogspace) which is increasingly valuable as a source of information. A comprehensive understanding of weblogs and blogspace is likely to prove critical, and may also offer useful insights into such development as the behavior of social networks or information diffusion. On the other hand, the increasing amount of choices available on the Web has made it increasingly difficult to find what one is looking for. Three methods are commonly applied to help web users in locating their desired items: search engines, taxonomies and, more recently, recommender systems [2].

The problem we are addressing in this paper is to design an algorithm to recommend weblog users what other weblogs they may enjoy reading. To do so, we assume that each user has a set of *favorite weblogs* and we want to identify a set of *similar* weblogs to the favorite weblogs. The weblog recommender systems discussed in this paper can be applied to any social network in which the relevance and relations among nodes of the network are captured by links among nodes of a graph. In particular, these link-based methods are useful for social networks with high average degree.

1.1 Our Contribution

In this paper, we design, implement, and compare algorithms for recommender systems for social networks based on their link structure, and evaluate our algorithms on a data set of weblogs. We first crawl a set of weblogs, and constructed the link structure among them as a weblog graph. We use ideas from spectral clustering algorithms [8] and a variant of local clustering algorithm based on refined random walks [14] and personalized PageRank algorithms [4] for our recommender system. In particular,

- We extend the truncated random walk algorithm to approximate the personalized PageRank vector [14, 4] from undirected graphs to directed graphs. To avoid accumulating probabilities on sinks in directed graphs, we design a non-uniform random walk as follows: we find strongly connected components of the directed weblog graph, and change the restarting probability of the random walk based on the position of each node in the structure of the strongly connected component decomposition. This will result in a non-uniform random walk for directed graphs that gives better results compared to performing a uniform random walk on the directed graph. The details of this algorithm can be found in Section 3.1.
- We define a spectral algorithm in which we calculate a *similarity metric* among nodes of the weblog graph based on the eigenvalues and eigenvectors of a normalized adjacency matrix of the weblog graph. For a set of favorite weblogs, this algorithm outputs the closest nodes to the favorite weblogs according to the similarity metric. The details of this similarity metric can be found in Section 4.
- We analyze the structure of the weblog graph for its average degree, and for its connected and strongly connected components. We then implement and evaluate the following four algorithms: a spectral algorithm applied to two candidate sets from the directed and undirected graph, and a truncated random walk algorithm to approximate the personalized PageRank vector on directed and undirected graphs. We compare these algorithms in terms of their precision and recall and their running time and conclude the paper by some insights from our comparison. In particular, our experimental results justify a link-based recommender system for weblogs and other social networks with high average degree. Our experimental results can be found in Section 5.

1.2 Related Work

Recommender systems have been extensively studied recently [2, 3, 11]. Recommender systems are closely related to clustering algorithms. Different clustering algorithms for large data sets have been proposed. In this paper, we use ideas from spectral clustering, and clustering based on random walks. A good survey of spectral clustering can be found in a recent paper by Verma and Meila [7]. They have studied several spectral algorithms in their paper and also compared their performance. In particular, they have studied the algorithms by R. Kannan, S. Vempala, A. Vetta [9], and an algorithm by J. Shi, J. Malik [8]. In the context of clustering algorithms based on random walks, the paper of Spielman and Teng [14] introduces a fast algorithm for local partitioning based on refined random walks. Also, a recent paper by Andersen, Chung and Lang [4] gives a local partitioning algorithm using the personalized PageRank.

Recommender systems are also related to ranking algorithms in that we need to rank the recommended items in some order. In this paper, we use ideas from the PageRank Algorithm that has been developed by Page, and Brin [5]. Another important variant of this algorithm is the personalized PageRank which was introduced by Haveliwal [15],

and has been used to provide personalized search ranking and context-sensitive search.

2. PRELIMINARIES

In this section, we define the terms that are used throughout the paper.

Weblog Graph: Nodes of the weblog graph are the set of weblogs. There is a directed edge from node w to node u if there exists a link from weblog w to weblog u . As a result, the weblog graph is a directed graph. We also study the undirected version of the weblog graph in which we make all edges of the directed graph undirected. In other words, we make the graph undirected by putting the backward edges in the graph.

Cut in a graph: A cut $(S, V(G) - S)$ of an undirected graph $G = (V, E)$ is the set of edges from S to $V(G) - S$ [13]. For two sets $A, B \subseteq V(G)$, $\text{Cut}(A, B)$ denote the set of edges from A to B .

Volume of set A : Let $\text{deg}(v)$ to be the degree of node v . Then, the volume of set A is denoted by $\text{Vol}(A) = \sum_{v \in A} \text{deg}(v)$.

Sparsity for two sets $A, B \subseteq V(G)$ is defined as $\text{Sparsity}(A, B) = \frac{|\text{Cut}(A, B)|}{\text{Vol}(A)\text{Vol}(B)}$ [7].

Connected Component: A connected component of an undirected graph is a maximal set of vertices such that for any two vertices u and v in the set, there is a path from u to v or from v to u .

Strongly Connected Component: A strongly connected component of a directed graph is a maximal set of vertices such that for any two vertices u and v in the set, there is a path from u to v and from v to u .

Random Walk: Given a graph and a starting point, we select a neighbor of it at random, and move to this neighbor, then we select a neighbor of this point at random and move to it etc.. The (random) sequence of points selected this way is a random walk on the graph [12].

Uniform Random Walk: A uniform random walk is a random walk in which at each step we go to each neighbor with the same probability [12].

Eigenvalue and Eigenvector: Let A be a linear transformation represented by a matrix A . If there is a vector $X \in R^n \neq 0$ such that $AX = \lambda X$ for some scalar λ , then λ is called the eigenvalue of A with corresponding (right) eigenvector X .

2.1 Recommender Systems

We assume that each user has a set of favorite items. Favorite items are the items which are already bought, used or chosen. The goal of a recommender system is to recommend a set of items, similar to the set of favorite items.

Recommender systems for weblogs can be designed based on the similarity among the content of the weblogs, the link structure between different weblogs, and the comments posed by the weblog owners. Here we briefly explain these three approaches separately.

1. Content based recommender systems: Based on the frequent words inside the posts and their titles, we can come up with a set of subject features for each weblog. Score of each feature in each weblog is determined by the frequency of the words related to this feature. We

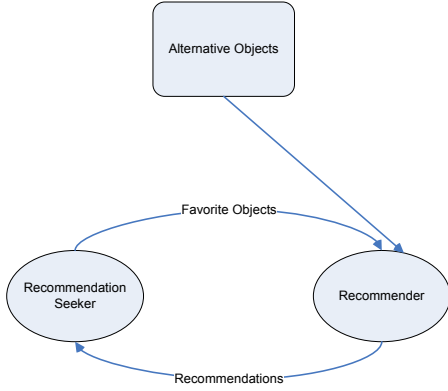


Figure 1: Recommendation System Model

can view this scoring system of features for weblogs as a ranking system and use this scoring to design a content based recommender system. In order to implement this recommender system we can use one of the known algorithms either model-based or memory-based algorithms. Many such algorithms are described in [1].

2. Link-based recommender systems: We can build the weblog graph assuming the weblogs as nodes and the links between weblogs as edges of this graph. Clustering this graph can help us reveal similarities among weblogs and as a result, it can be used to design a recommender system. Note that the outgoing links from weblogs can be transferred by users to find the similar and related weblogs, but the main advantage of constructing the weblog graph is discovering the similarities between the incoming links for different weblogs. This idea can be used in combination with the content-based recommender systems and improve its outcome. We can use the known clustering algorithms such as clustering based on random walks [14, 4], spectral clustering [7], or any type of well-connected components to reveal this information.
3. Comment-based recommender systems: Instead of constructing the weblog graph, only based on the links between weblogs, we can build the graph based on the posted comments inside the weblogs. The idea is to put an edge from weblog A to weblog B, if the owner of weblog A put a comment (or several comments) for the posts of weblog B. We can then cluster this graph which reveals the relation among weblogs.

3. LOCAL PARTITIONING ALGORITHMS BASED ON RANDOM WALKS

In this section, we present a link-based recommender system based on computing the personalized PageRank of favorite nodes using refined random walks in a social network. In this algorithm, we consider an undirected, unweighted graph $G(V, E)$, where V is the vertex set, E is the edge set, n is the number of vertices and $\text{deg}(v)$ is the degree of vertex v . The algorithm uses a restarting probability distribution vector \vec{r} . For example, in the PageRank algorithm [5], the

distribution is uniform over all nodes. Also, for the personalized PageRank [4, 5], the vector is one at a single vertex and zero elsewhere. Moreover, if we are given a set S of k weblogs for which we want to find relevant weblogs, we can set the restarting vector \vec{r} as a vector with value $\frac{1}{k}$ on the k vertices corresponding to the set S of weblogs or the user can distribute 1 over k favorite weblogs. Using this restarting probability distribution, we claim that the weblogs that have the largest PageRank values in the following algorithm are more relevant to the weblogs in set S .

Given the restarting vector \vec{r} , we can run a PageRank algorithm with this restarting probability distribution and report the nodes with high PageRank as the output. The algorithm also uses a restarting probability $p = 0.15$, which is the probability that we restart the random walk from vector \vec{r} . In fact, we use a variant of random walk which is called the *lazy random walk*. In this variant, at each vertex with probability, ℓ , we remain in the same vertex. Since we are interested in the similarity to our favorite items, using lazy random walks is more appropriate.

We construct the transition matrix A that corresponds to the PageRank random walk with restarting probability p and restarting vector \vec{r} . $A[i, j]$ is in fact the probability that we will move to vertex j in the graph given that we are in vertex i , that is, for an edge (i, j) , we have:

$$A[i, j] = \frac{(1 - p - \ell)}{\text{deg}(i)} + p\vec{r}(j),$$

for $i = j$,

$$A[i, j] = \ell + p\vec{r}(j),$$

and for a non-edge pair (i, j) :

$$A[i, j] = p\vec{r}(j),$$

First, we give a rough description of the algorithm. Given the restarting vector \vec{r} , the restarting probability p , the laziness probability ℓ , and the transition matrix A , the algorithm is as follows:

1. $\vec{v} = \vec{r}$. (We start from vector \vec{r} .)
2. $\vec{w} = A\vec{v}$.
3. while ($\text{dist}(\vec{w}, \vec{v}) > 0.0001$) do
 - (a) $\vec{v} = \vec{w}$.
 - (b) $\vec{w} = A\vec{v}$.
4. Sort weblogs in the non-increasing order of the value $\vec{v}(i)$ and output them in this order. If some of the weblogs are already in the list of favorite weblogs or their immediate neighbors, do not output them.

$\text{dist}(\vec{w}, \vec{v})$ in this algorithm is the ℓ_2 distance between \vec{w} and \vec{v} .

We would like to improve the running time of the above algorithm to find the b most relevant nodes to a given set S of k favorite nodes in the graph. In particular, we want to design an algorithm whose running time depends on the number b of relevant nodes in the output, and not the size of the whole graph. In other words, we need to design a local way of approximating the personalized PageRank vector of a set of nodes. There are two ways to perform such a task:

Truncating Probability Vectors. The idea of truncating the probability vectors has been developed by Spielman and Teng [14] for local partitioning based on random walks. In this algorithm, after each step of the random walk, we change the entries of the probability distribution vector that are less than a threshold to zero, and then normalize the vector. A natural threshold to use is $\alpha = \frac{\epsilon}{2b}$ where $0 < \epsilon < 1$.¹ Given the restarting vector \vec{r} , the number of weblogs that we would like to recommend b , a threshold for pruning the resulting vectors α , and a transition matrix A , the algorithm is as follows:

1. $\vec{v} = \vec{r}$ (i.e, we start from vector \vec{r} .)
2. $\vec{w} = A\vec{v}$.
3. Let q be the number of nodes with positive scores in w .
4. while ((number of nodes with positive scores $\leq b + q$) and ($dist(v, w) > 0.0001$)) do
 - (a) $\vec{v} = \vec{w}$.
 - (b) $\vec{w} = A\vec{v}$.
 - (c) for each $i \in V(G)$ do
 - i. if $\vec{w}[i] \leq \alpha$ then $\vec{w}[i] = 0$.
 - (d) normalize(\vec{w}).

In our experiments, we implemented the above algorithm. However, for the sake of completeness, we present an alternative way to perform a locally fast random walk.

Sequentially Pushing Probability from Nodes. The idea of this algorithm has been developed by Andersen, Chung, and Lang [4] for approximating the personalized PageRank vector. The algorithm maintains two vectors \vec{p} and \vec{q} that are initialized to $\vec{0}$ and the restarting vector \vec{r} respectively. The algorithm then applies a series of push operations moving probability from \vec{q} to \vec{p} . Each push operation takes the probability from \vec{q} at a single vertex u , and moves an α fraction of this probability to $p(u)$, and distributes the $1 - \alpha$ fraction of $q(u)$ within \vec{q} by applying one step of the lazy random walk from u . The algorithm is summarized as the following:

1. Let $\vec{p} = \vec{0}$ and $\vec{q} = \vec{r}$.
2. While $r(u) \geq \epsilon \deg(u)$ for some node u ,
 - (a) Pick any vertex u where $r(u) \geq \epsilon \deg(u)$.
 - (b) Update p and q :
 - i. $p(u) = p(u) + \alpha r(u)$.
 - ii. For each outgoing neighbor v of node u ,

$$r'(v) = r(v) + \frac{(1-\alpha)r(u)}{2 \deg(u)}.$$
 - iii. $r(u) = (1 - \alpha) \frac{r(u)}{2}$.

¹In order to avoid infinite loop, we need to have $\alpha < \frac{1}{b}$.

3.1 Random Walk on Directed Graphs

The local partitioning algorithms described in Section 3 are used for undirected graphs and can be applied to the undirected version of the weblog graph. The original weblog graph is a directed graph. If we apply the ideas from the aforementioned algorithms on directed graphs, one issue is that if the directed graph has a sink, most of the probability measure will go to the sink. In order to apply the ideas from those algorithms on directed graphs, we need to find the strongly connected components(SCC) of the directed graph, and change the restarting probability based on the place of the nodes in strongly connected components. In this algorithm, we consider a directed, unweighted graph $G(V, E)$, where V is the vertex set, E is the edge set, n is the number of vertices, and $\text{outdeg}(v)$ is the degree of vertex v . Using a linear-time algorithm by Tarjan [6], we first find the strongly connected components of this graph. For a node $v \in V(G)$, let $SCC(v)$ be the strongly connected component containing the node v . We can also find connected components of the undirected graph corresponding to the directed graph. This can be found using a linear-time breadth-first-search algorithm. For a node $v \in V(G)$, let $CC(v)$ be the connected component containing the node v . It follows that $SCC(v) \subseteq CC(v)$. For a subset S of nodes, let $SCC(S) = \cup_{v \in S} SCC(v)$ and $CC(S) = \cup_{v \in S} CC(v)$.

Given a set of favorite nodes S , we partition all nodes into five categories:

Category 1. Nodes in the $SCC(S)$.

Category 2. Nodes that have a directed path to $SCC(S)$.

Category 3. Nodes that have a directed path from $SCC(S)$.

Category 4. All the other nodes in $CC(S) - SCC(S)$.

Category 5. Nodes in $V(G) - CC(S)$.

Let $Category^S(i)$ be the set of nodes of category i is given a set of favorite nodes S .

The idea for directed graphs is to change the restarting probability p based on the category of a node for set S . As discussed in Section 3, the set of favorite weblogs S is the support of the restarting vector r . As a result, starting from a set of favorite weblogs S , the probability of getting to any node of category 5 in a random walk of Algorithm PPR is zero. If we perform the random walk on a directed graph, the probability of getting to any node of category 2 and 4 is also zero. Nodes of category 3 are sinks of the random walk, since there is no path from them to $SCC(S)$. To decrease their rank in the , we set the restarting probability of a node $v \in Category^S(3)$ to a larger number, say $p(v) = 0.5$. For all the other nodes $v \in Category^S(1)$, we set the restarting probability to $p(v) = 0.15$. We also put more probability mass on edges to the same strongly connected component. Let $w : V(G) \rightarrow V(G)$ be a weight function defined as follows: $w(u, v) = 0$ if $(u, v) \notin E(G)$. Otherwise, we let $w(u, v) = 1$ if $v \in SCC(u)$ and $w(u, v) = 0.3$ if $v \notin SCC(v)$.

As a result, we construct the transition matrix A that corresponds to our non-uniform random walk with restarting probability p and restarting vector \vec{r} . $A[i, j]$ is the probability that we will move to vertex j in the graph given that we

are in vertex i , that is, for an edge (i, j) , we have:

$$A[i, j] = \frac{(1 - p(i) - \ell)w(i, j)}{\sum_{t:(i,t) \in E(G)w(i,t)} + p(i)\bar{r}(j)},$$

for $i = j$,

$$A[i, j] = \ell + p(i)\bar{r}(j),$$

and for a non-edge pair (i, j) :

$$A[i, j] = p(i)\bar{r}(j).$$

Given the above transition matrix, we can run the truncated random walk algorithm discussed in Section 3 on the directed graph to compute the approximate personalized PageRank of the nodes.

4. A SPECTRAL METHOD

We use the intuition behind the spectral clustering to compute a *similarity metric* for the purpose of our recommender system. Recall that the sparsity for two sets $A, B \subseteq V(G)$ is defined as $\text{Sparsity}(A, B) = \frac{|\text{Cut}(A, B)|}{\text{Vol}(A)\text{Vol}(B)}$. A goal of clustering is to find cuts for which the sparsity is small. Spectral clustering is suitable for clustering graphs in which the maximum sparsity is not large. Spectral clustering is considered to be a hierarchical and global clustering. Hence, in order to use spectral clustering for our system, we need to change the current known algorithms for spectral clustering. The idea is to use the eigenvectors of a normalized adjacency matrix to evaluate the distance between nodes of a graph. In order to recommend a set of weblogs relevant to a favorite weblog w , we can output the weblogs that have small distance to weblog w according to a similarity metric computed based on the eigenvectors of the largest eigenvalues. Intuitively, each eigenvector of a large eigenvalue represents one type of clustering the data into two clusters. The importance of an eigenvector is directly related to its corresponding eigenvalue, i.e., a larger eigenvalue indicates a better partitioning of the graph.

There are several variants of the spectral clustering algorithms in the literature. A good survey on this topic is by Verma, and Meila [7]. They categorize spectral algorithms into two major sets: (i) *Recursive Spectral*, in which, we partition data into two sets according to a single eigenvector and by recursion we generate the specified number of clusters, or (ii) *Multway Spectral*, in which, we use the information hidden in multiple eigenvectors in order to directly partition data into the specified number of clusters. In particular, our algorithm is based on the ideas for the recursive spectral algorithm proposed by Semi and Malik [8], Kannan, Vempala, and Vetta [9], and the multiway spectral algorithm of Ng, Jordan, and Weiss [10].

Let A be the adjacency matrix of the weblog graph, and D be a diagonal matrix such that $D[i, i] = \text{deg}(i)$ for a node i . Given a favorite weblog w for which we want to find similar weblogs, our algorithm is as follows:

1. Calculate $P = D^{-1}A$
2. Let $1 = \lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n$ be the eigenvalues of P , compute v^2, v^3, \dots, v^t which are the eigenvectors corresponding to $\lambda_2, \lambda_3, \dots, \lambda_t$.

3. Find a set of weights c_2, \dots, c_t for each eigenvector. c_i shows how valuable or important vector v^i is for clustering purposes. Intuitively, if $\lambda_i \leq 0$, $c_i = 0$, and for $\lambda_i > 0$, $c_i = f(\lambda_i)$ where $f(x)$ is an increasing convex function, for example $f(x) = x^2$, or $f(x) = e^x$.
4. For any weblog j and any vector v^i , let the difference $d_i(j) = |v^i(j) - v^i(w)|$.
5. Sort the vertices in the non-decreasing order of the value $q_j = \sum_{2 \leq i \leq t} c_i d_i(j)$ and output the vertices in this order.

The vector q is the distance measure to node w . Considering this distance measure for all nodes results in a similarity metric over all pairs of nodes.

Since computing the eigenvalue and eigenvectors require matrix multiplication, the running time of the above algorithm is at least as bad as the matrix multiplication algorithm².

However, we could approximate the eigenvalues and eigenvectors of the graph and use them to define the metric. In our experiments, we improve the running time of this algorithm by computing a *candidate set* of 1000 weblogs related to our favorite weblog, and then use the spectral algorithm to rank weblogs in the candidate set. In order to find this candidate set, we could use the content information of weblogs or use some local partitioning algorithm that finds all weblogs that are relevant to a particular weblog. In our experiments, we compute the truncated random walk algorithm that computes the approximate personalized PageRank of each weblog with respect to the favorite weblog. As a candidate set of weblogs, we output 1000 weblogs with highest approximate personalized PageRank with respect to the favorite weblog. Then we can find the above metric based on the eigenvalue and eigenvectors of the induced graph of the candidate set to rank the nodes of this set.

5. EVALUATION

Evaluating recommender systems and their algorithms is inherently difficult for several reasons. First, different algorithms may be better or worse on different data sets. Many collaborative filtering algorithms have been designed specifically for data sets where there are many more users than items. Such algorithms may be entirely inappropriate in a domain where there are many more items than users. Similar differences exist for ratings density, ratings scale, and other properties of data sets. The second reason that evaluation is difficult is that the goals for which an evaluation is performed may differ. Finally, there is a significant challenge in deciding what combination of measures to use in comparative evaluation [11].

5.1 Implementation

In order to evaluate the performance of the algorithms, we implemented a crawler and made a dataset. The crawler is a focused crawler meaning that it only crawls weblogs which are not in the weblogs' domain. We constructed the weblog graph for the dataset and performed four algorithms on it:

²The best known algorithm for matrix multiplication is $O(n^\alpha)$ where $\alpha = 2.37$

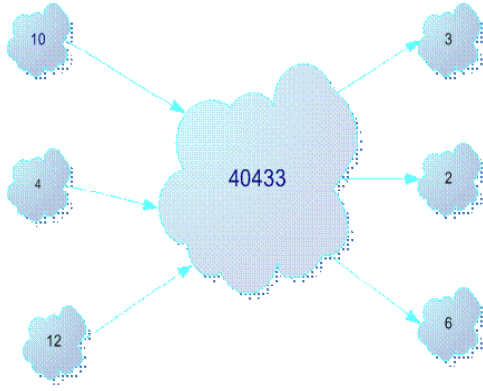


Figure 2: An abstract model of our weblog graph: strongly connected components.

1. Approximate personalized PageRank on directed graphs (DPPR).
2. Approximate personalized PageRank on undirected graphs (PPR).
3. Spectral algorithm applied on the a candidate set from DPPR (SDPPR).
4. Spectral algorithm applied on the a candidate set from PPR (SPPR).

As discussed in Section 3, we have two options for implementing the local computation of approximate personalized PageRank: the truncated random walk, and pushing probability from nodes. We used the truncated random walk algorithm for our implementation and achieved improved running time. Since the spectral algorithm is not based on a local computation, we run the PPR and DPPR algorithms to find a set of 2000 candidate nodes, and then compute the spectral metric for this set of nodes, and rank them based on the distance of these nodes in this metric. In order to find the set of candidate nodes to feed to this spectral algorithm, one can use other types of content-based algorithms as well.

5.2 Metrics

For the system we are going to develop, we adapt two metrics defined in information retrieval, *precision* and *recall*. In this case we define precision and recall respectively as follows:

$$\text{Precision} = \frac{|\text{Recommended weblogs} \cap \text{Favorite weblogs}|}{|\text{Recommended weblogs}|},$$

$$\text{Recall} = \frac{|\text{Recommended weblogs} \cap \text{Favorite weblogs}|}{|\text{Favorite weblogs}|}.$$

5.3 Experimental Results

In order to perform such evaluation, we omit a portion of links from the list of outgoing links of a weblog and reconstruct the graph without that information. Then, we apply the algorithm on the new graph and see how many omitted links are rediscovered, then we calculate the metrics.

Our dataset contains 120000 weblogs, which have links to about 4000000 other weblogs. The average degree of the

weblog graph is 50. This average degree is more than the typical average degree of nodes of other social networks like the web graph. This implies that there is more information hidden in the link structure of this graph compared to other social networks. As we will see, this might be the main reason that our algorithm performs very well on this graph. Moreover, as we expected the weblog graph has a large strongly connected component containing 40433 nodes and many small strongly connected components.

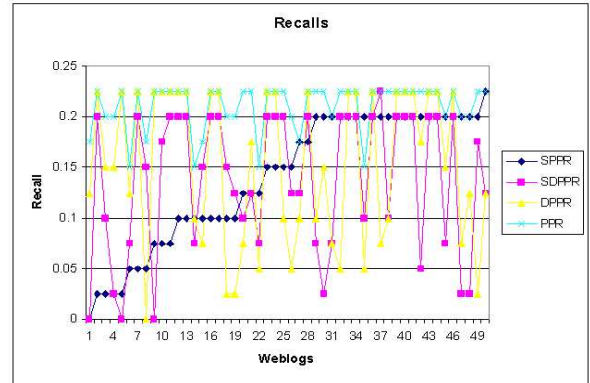


Figure 3: Recalls.

In order to have meaningful experiments, we select 50 high-degree weblogs at random. The degree of each of these weblogs were at least 50, and we removed 10 to 15 outgoing neighbors of each of these 50 weblogs. The recalls and precisions calculated for these 50 randomly chosen weblogs are plotted in Figures 3 and 4 respectively. The average of recall and precision for all four algorithms are also plotted in Figure 5. As it can be seen, all the four algorithms have acceptable performance. The performance of these four algorithms is ordered as follows: personalized PageRank algorithm on undirected graphs, personalized PageRank algorithm on undirected graphs, Spectral algorithm applied on the a candidate set from DPPR, Spectral algorithm applied on the a candidate set from DPPR. These experiments justify the use of link structure for a recommender system for weblogs, and perhaps other social networks with high average degree. Moreover, it proves the applicability of local partitioning algorithms to estimate the personalized PageRank algorithm.

Running Time. Both PPR and DPPR algorithms are based on local computation of approximate personalized PageRank vectors and run very fast. We implement these two algorithms in C. In comparison, the running time of the algorithm on directed graphs is faster. The reason is that the outdegree of vertices is smaller than its degree in the undirected graph and as a result we deal with larger vectors in PPR.

The time to construct the similarity metric based on the spectral method for the whole graph is much worse than the running time of the local partitioning algorithms, however, since we apply the algorithm on a candidate set of 1000 nodes and then perform the spectral method on the candidate set, the running time is comparable. The spectral algorithm is implemented in Matlab.

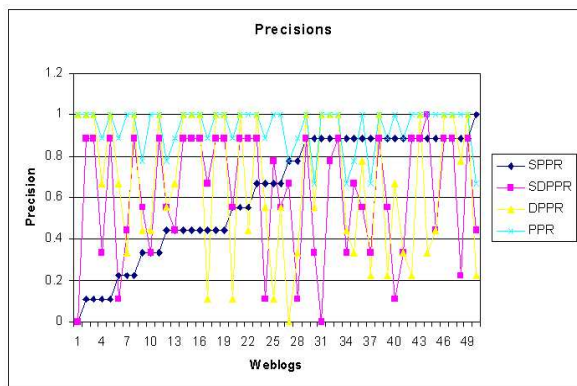


Figure 4: Precisions.

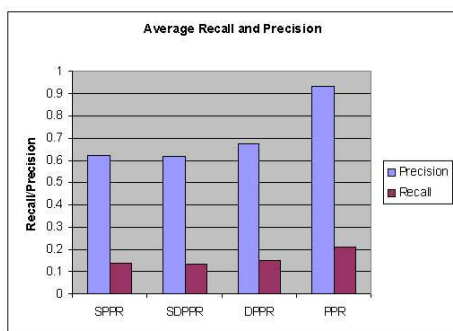


Figure 5: Comparing average of recalls and precisions.

6. CONCLUSIONS

In this paper, we develop two main ideas for a recommender system based on the link structure of the weblog graph. We examine these two ideas on directed and undirected graphs. One idea is to use the personalized PageRank vector to capture the relevance of nodes on a graph and apply the local truncated random walk algorithms to approximate this vector. We also extend the random walk to a refined random walk on directed graphs. The other idea is to introduce a metric based on the spectral methods that can be used to rank the relevant nodes in a network. In order to evaluate the performance of the algorithms, we construct a dataset and compare the precision and recall of these methods on a sample subset of this dataset.

Our experimental results show that the performance of all algorithms are acceptable for weblogs. The main reason for this good performance is the high average degree of the weblog graph which indicates that one can achieve a reasonable performance for a recommender system for weblogs by only using the link information of the weblog graph. This observation implies that for social networks with high average degree using the link structure for a recommender system is a realistic approach.

The local approximate personalized PageRank algorithm has a better performance compared to the spectral method for our data set. The running time of the personalized PageRank algorithm is also much better than the spectral method that we used. The results of the spectral method

are mainly used as a base for comparison, and justify the use of the random walk algorithm. Between the two personalized PageRank algorithms on directed and undirected graphs, the algorithm for directed graphs run faster, but the performance of the algorithm for undirected graphs is better.

7. REFERENCES

- [1] <http://www.blogpulse.com> visited Dec. 2006.
- [2] J. Parsons, P. Ralph, K. Gallagher. Using viewing time to infer user preference in recommender systems. AAAI Workshop in Semantic Web Personalization, San Jose, California, July,2004.
- [3] P. Resnick, H. Varian. Recommender Systems. Communications of the ACM, vol 40, 56-58, 1997.
- [4] R. Andersen, F. Chung, K. Lang. Local Graph Partitioning using PageRank Vectors, Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06),Pages: 475 - 486.
- [5] S. Brin, L. Page,R. Motwani, T. Winograd. The PageRank citation ranking: Bringing order to the web., Technical report, Stanford Digital Library Technologies Project, 1998.
- [6] R. E. Tarjan. Depth-first search and linear graph algorithms, SIAM Journal on Computing, 1(2):146-160, 1972.
- [7] D. Verma, M. Meila. A comparison of spectral clustering algorithms., technical report UW-cse-03-05-01, University of Washington.
- [8] J. Shi, J. Malik. Normalized cuts and image segmentation, IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(8):888-905, 2000.
- [9] R. Kannan, S. Vempala, A. Vetta. On clusterings-good, bad and spectral, Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS'2000),Pages: 367-377.
- [10] A. Ng, I. Jordan, Y. Weiss. On Spectral Clustering: Analysis and an algorithm, Advances in Neural Information Processing Systems, 14, pages 849-856.
- [11] J.L. Herlocker, J.A. Konstan, L.G. Terveen, J.T. Riedl. Evaluating Collaborative Filtering Recommender Systems, ACM Transactions on Information Systems (TOIS), 2004.
- [12] L. Lovasz. Random walks on graphs: A survey. January 1993.
- [13] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein. Introduction to Algorithms (Second Edition) . 2001.
- [14] D.A. Spielman, S. Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. ACM STOC-04, pages 81-90, New York, NY, USA, 2004. ACM Press.
- [15] T.H. Haveliwala.Topic-sensitive PageRank: A context-sensitive ranking algorithm for web search. IEEE Trans. Knowl. Data Eng., 15(4):784-796, 2003.