Using *Random Walks* for Mining Web Document Associations

K. Selçuk Candan[2⋆] and Wen-Syan Li[1]

[1] C&C Research Laboratories, NEC USA, Inc., MS/SJ10, San Jose, CA 95134,USA
{candan, wen}@ccrl.sj.nec.com
[2] Computer Sci, and Eng. Dept., Arizona State University, Tempe, AZ 85287, USA
candan@asu.edu

Abstract. World Wide Web has emerged as a primary means for storing and structuring information. In this paper, we present a framework for mining implicit associations among Web documents. We focus on the following problem: "For a given set of seed URLs, find a list of Web pages which reflect the association among these seeds." In the proposed framework, associations of two documents are induced by the connectivity and linking path length. Based on this framework, we have developed a *random walk*-based Web mining technique and validated it by experiments on real Web data. In this paper, we also discuss the extension of the algorithm for considering document contents.

1 Introduction

In traditional information retrieval field, in order to determine the association between a given set of documents, keyword vectors that represent the contents of these document are compared. A major difference between Web pages and textual documents is that Web pages have links connecting to other related pages. When an author prepares a Web document, he/she would put contents on each page while linking related information together using anchors, to create a document spanning multiple pages. Thus, Web structures can be used as hints to derive document association. Existing approaches for finding Web document associations include the *companion* and *co-citation* algorithms proposed by Dean and Henzinger[1] and the Netscape algorithm[2] used to implement the `What's Related?` functionalities. In this paper, we are interested not only in deriving document associations, but also in inducing the reasons why they are associated. We focus on the problem "for a given two seed URLs, find a list of Web pages which reflex the association such two seed URLs."

Example 1. In Figure 1, we show a subset of links between two personal Web pages `W.Li` and `D.Agrawal`. The purpose of each link is indicated in the link label. The associations between `W.Li` and `D.Agrawal` are implicitly expressed in the Web structure connecting `W.Li` and `D.Agrawal` even though this structure is created independently by many individuals. Below, we enumerate some associations that can be derived from this graph, and some possible interpretations:

⋆ This work was performed when the author visited NEC, CCRL.

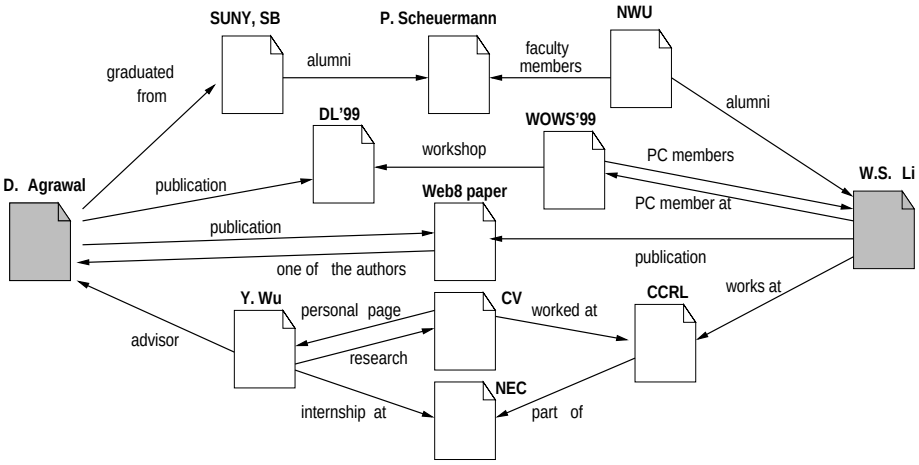


Fig. 1. Link structure connecting and associating the Web pages of `W.Li` and `D.Agrawal`

• `Web8 paper` page appears in a path of distance of 2 connecting the two pages. Therefore, `W.Li` and `D.Agrawal` may be associated due to a co-authored paper.
• `Y.Wu` page is on two paths each of distance 4. `W.Li` and `D.Agrawal` may be associated due to the fact they both supervised `Y.Wu`.
• `WOWS'99` and `ACM DL'99` pages appear on a single path of distance 3. `W.Li` and `D.Agrawal` are participating in the same conference.
• `P.Scheuermann`, `D.Agrawal`, `NWU`, `W.Li`, and `SUNY` pages appear on a single path of distance 4. `D.Agrawal` and `W.Li` may be associated due to some people related to `SUNY, SB` or due to an alumni relationship.

Obviously, the links connecting these pages are not intended to express such associations and the authors are not coordinated to make the link semantics consistent across the Web. However, we can use the following two intuitions: (1) *Path length:* Pages on a shorter path between the two pages in the example are stronger indicators than others to reflect why these pages are associated; and *Connectivity:* Pages which appear on more paths are stronger indicators than others to reflect why the two pages are associated.

Note that a page with a higher connectivity (i.e. more incoming links and outgoing links) is more likely to be included in more paths; consequently, such a page is more likely to be ranked higher according to the above criteria. This is consistent with the principle of topic distillation[3, 4]. On the other hand, to address the associativity problem, we also need to consider the distance between a page and seed URLs to account for the first intuition. Thus, a page with a high connectivity but far away from the seed URLs may be less significant to represent the seed URLs after associations than a page with low connectivity but close to the seed URLs. A page which satisfies both criteria (i.e. near seed URLs and with high connectivity) is a good representative for the association.

Based on the motivation and intuitions, we present a novel framework for mining associations among Web documents using information *implicitly* reflect-

ed in the links connecting them. We develop a Web mining technique, based on a *random walk algorithm*, which considers document distances by link and connectivity. In the concluding remarks, we also briefly show the algorithm can be extended to be specific content focused (e.g. finding why the `W.Li` and `D.Agrawal` are associated with respect to the *PowerBookmarks* project).

2 *Random Walks* Algorithm

In this section, we introduce the modeling of the framework and the algorithm.

2.1 Modeling

Let us assume that we are interested in mining the associations of a set, $\mathcal{S} = \{s_1, \ldots, s_n\}$, of seed Web pages (or *snodes*). The mining task is to find a set $Ref(\mathcal{S})$, of pages that best induce (or reflect) the association among a given set of *snode*s. We denote such pages inductive Web pages (or *inodes*). For ease of presentation, we start with the case where there are only two seed pages for association mining. The cases where $\mathcal{S}$ contains more than two pages is discussed in Section 2.3.

Let the Web be modeled as a directed graph, $G(V, E)$, and let the two seed pages in $\mathcal{S}$, defined as *snode*, correspond to vertices v_a and v_b in V. Let us also assume that we want to find an *inode* page (or vertex) within a radius of d from v_a or v_b. Note that the choice of d is application dependent. If *progressive* results are required, d can be incremented starting from 1, refining the results at each step, until either the process times out or an acceptable *inode* is located.

Links have been used in many fields to associate documents. They can be categorized into four types: connectivity, co-citation, social filtering, and transitivity[4]. Since the association mining problem as formulated in this paper is symmetric, we do not differentiate these. Consequently, we use an undirected graph, $G^u(V, E^u)$, to model the Web. Assuming that we are given a radius d, we define the relevant neighborhood, $G^N(V^N, E^N)$, of $G^u(V, E^u)$, as the set of vertices, $V^N = V_{G^u}(v_a, v_b, d)$, that are reachable either from v_a or v_b in d edge traversals: $(\forall v_i \in V_{G^u}(v_a, v_b, d)\ \ reachable_{G^u}(v_a, v_i, d) \vee reachable_{G^u}(v_b, v_i, d))$ Note that without loss of generality, we will assume that the graph, G^N, is connected.

To derive metrics for *Inode* selection, one straight forward candidate metric, that adjusts connectivity scores by distance, for *inode* selection would be

$$score(v) = \sum_{p \in paths(A,B,v)} \frac{1}{length(p)},$$

where $paths(A, B, v)$ is the set of (simple) paths between the seeds, A and B, that pass through a candidate *inode*, v, and $length(p)$ is the length of the path.

Note that, although it merges the two required structural criteria, this metric has two major disadvantages preventing its use in association mining. **(1)** First, its calculation may require the enumeration of all paths in the graph, which may require exponential time with respect to the size of the graph, and **(2)** although the maximum length of the paths grows linearly with the number of vertices in

1. $\mathcal{V} = \emptyset$;
2. For each $v_i \in V^N$, create a new node v_i' and insert it in $\mathcal{V}$;
3. $\mathcal{E} = \emptyset$;
4. For each $e_k = \langle v_i, v_j \rangle \in E^u$ such that both v_i and v_j are in V^N
   (a) create two directed edges $e'_{2\times k} = \langle v_i', v_j' \rangle$ and $e'_{2\times k+1} = \langle v_j', v_i' \rangle$ and insert them in $\mathcal{E}$;
5. For all vertices $v_i' \in \mathcal{V}$, let
   (a) $sdist(v_i', v_a')$ be the shortest distance, in G^N, between v_i' and the vertex, v_a', corresponding to v_a, and
   (b) $sdist(v_i', v_b')$ be the shortest distance, in G^N, between v_i' and the vertex, v_b', corresponding to v_b
   (c) $penalty(v_i') = sdist(v_i', v_a') + sdist(v_i', v_b')$.
   (d) For all vertices $v_i' \in \mathcal{V}$ and for all $\langle v_i', v_j' \rangle \notin \mathcal{E}$, $\mathcal{T}[j, i] = 0.0$;
   (e) For all vertices $v_i' \in \mathcal{V}$ solve the following set of linear equations:

$$L(v_i') = \left\{ \sum_{\langle v_i', v_j' \rangle \in \mathcal{E}} \mathcal{T}[j, i] = 1.0 \right\} \bigcup$$

$$\left\{ \mathcal{T}[j, i] \times penalty(v_j') = \mathcal{T}[k, i] \times penalty(v_k') \;\middle|\; \langle v_i', v_j' \rangle \in \mathcal{E} \text{ and } \langle v_i', v_k' \rangle \in \mathcal{E} \right\}$$

Fig. 2. Algorithm for constructing a random walk graph

the graph, the number of paths grows exponentially as shown in our experiments. As a consequence, contrary to the intuition, the effect of the long paths (since their number is exponentially higher than the number of shorter paths) on the calculation of $score(v)$ is likely to be much larger than the effect of short paths.

2.2 Case 1: $\mathcal{S}$ contains Two Seed Pages

Consequently, instead of explicitly defining a metric, we will choose a set of random walk parameters that will implicitly capture the essence of these observations. For this purpose, we define and construct a random walk graph that reflects the required random walk parameters.

Definition 1 (Random Walk Graph). A random walk graph $\mathcal{R}(\mathcal{V}, \mathcal{E}, \mathcal{T})$ is a triple, where $\mathcal{V}$ is a set of vertices, $\mathcal{E}$ is a set of directed edges, and $\mathcal{T}$ is a $|\mathcal{V}| \times |\mathcal{V}|$ matrix where

- $\mathcal{T}[j, i]$ denotes the likelihood of moving to vertex v_i from vertex v_j.

Note that $\sum_{1 \le j \le |\mathcal{V}|} \mathcal{T}[j, i] = 1.0$ o

Algorithm 21 (Constructing a Random Walk Graph) Given an undirected neighborhood graph $G^N(V^N, E^N)$, two vertices v_a and v_b in V, and a radius d, we can construct a directed random walk graph $\mathcal{R}_{(v_a, v_b, d)}(\mathcal{V}, \mathcal{E}, \mathcal{T})$ using the algorithm presented in Figure 2. o

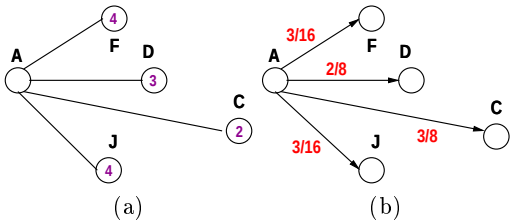


Fig. 3. (a) Penalty of each node and (b) transition values of each node

Description of the Algorithm for Constructing a Random Walk Graph Steps 1 and 2 of this algorithm insert the relevant vertices in the neighborhood into the random walk graph. Note that these two steps can be performed incrementally until a subgraph within a radius of d is explored. The next two steps use the undirected edges in the neighborhood graph to define two transitions (forward and backward) between the vertices in the random walk graph. These two transitions allow the random walk to proceed freely, back on forth, between the neighboring vertices of the graph.

Step 5, then, calculates a *penalty* for each node. This penalty term reflects the distance of each vertex from the seed vertices. Hence, for the case with two seeds, we define the penalty as the sum of shortest path distances between the given vertex and two seed vertices. We use the penalty to calculate the likelihood of each vertex being visited by the random walk process; more specifically, we calculate the transition probabilities of the edges in the graph using this term.

Since, by definition, a higher penalty means a greater distance from the seeds, it should yield a lower *association* score. Consequently, once the random walk process is at a vertex, v_i, it must proceed to a subsequent vertex, v_j, with a probability inversely proportional to v_js penalty. Furthermore, since the random walk will continue for an indefinite amount of time, the probability that the random walk process will leave vertex v_i (that is, it will proceed to one of its neighbors) must be equal to 1.0.

Example 2. Let us reconsider our example and focus on the portion of the graph shown in Figure 3(a), which depicts the vertex A, its four neighbors (F, D, C, and J), and the associated penalties calculated according to a distance metric (for the sake of simplicity, we omit the phase of penalty calculation). The following items reflect *some* of the facts about the transition probabilities of the edges leaving A:

- The sum of all such transition probabilities is equal to 1.0.
- Since the penalty of the vertex F is twice as much as the penalty of vertex C, the transition probability from A to F must be half of the transition probability from A to C.
- Since the penalty of the vertex D is $\frac{3}{2}$ times as much as the penalty of vertex C, the transition probability from A to D must be $\frac{2}{3}$ of the transition probability from A to C.

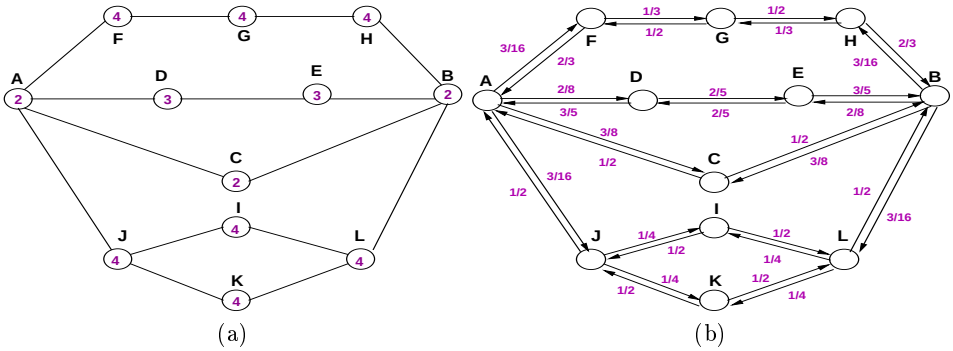


Fig. 4. (a) Penalty values, and (b) transition values for the example Web graph in Figure 1

Hence, we can calculate the transition values for the edges leaving A using the following set of constraints (as described in Step 5(e) of the algorithm):

$$\begin{aligned} \mathcal{T}[F,A] + \mathcal{T}[D,A] + \mathcal{T}[C,A] + \mathcal{T}[J,A] &= 1.0; \\ 4 \times \mathcal{T}[F,A] = 3 * \mathcal{T}[D,A]\ ; &\quad 3 \times \mathcal{T}[D,A] = 2 * \mathcal{T}[C,A]; \\ 2 \times \mathcal{T}[C,A] = 4 * \mathcal{T}[J,A]\ ; &\quad 4 \times \mathcal{T}[J,A] = 4 * \mathcal{T}[F,A]; \\ 4 \times \mathcal{T}[F,A] = 2 * \mathcal{T}[C,A]\ ; &\quad 4 \times \mathcal{T}[J,A] = 3 * \mathcal{T}[D,A]; \end{aligned}$$

Note that only the first four equations are enough to solve for all the unknowns. Figure 3(b) shows the transition values obtained by solving these constraints. o

Definition 2 (Convergence Vector). Given a random walk graph, $\mathcal{R}_{(v_a,v_b,d)}(\mathcal{V},\mathcal{E},\mathcal{T})$, t is called a convergence vector of $\mathcal{T}$ if $(t = \mathcal{T}t)$. o

Note that due to the structure of the transition matrix, such a convergence vector is guaranteed to exist. Intuitively, $t[i]$, describes the percentage of its time that a random walk process will spend in vertex $v[i]$ in a sufficiently long random walk. As we described earlier, *the higher this ratio, the better inode is the corresponding vertex. Consequently, we choose the inodes using their corresponding values in the convergence vector.*

Definition 3 (*Inode* Vertex). Given a graph $G(V,E)$, the *inode vertex* with respect to vertices v_a and v_b in G and a distance d, denoted as $inode_G(v_a, v_b, d)$ is a vertex $v_k \in V^N$ such that $t[k] = max\{t[i] \mid v'_i \in \mathcal{V}\}$. We also say that, if $t[i] > t[j]$, then v_i is more dominant than v_j. o

Example 3. Let us assume that Figure 4(a) shows a portion of a graph, G^u, where each shown vertex, v_i, is reachable from vertex A or B in 2 edges. The numbers shown in the vertices of the graph in Figure 4(a) are the corresponding distance penalties of the vertices. Figure 4(b), then, shows the corresponding random walk graph, $\mathcal{R}_{(A,B,2)}$. The transition values are shown as labels of the edges. The corresponding transition matrix $\mathcal{T}$ is also shown in Table 1(a).

Then, if we solve the linear equation $(I - \mathcal{T})t = 0$ (i.e. 12 variables and 13 constraints), we can find t as shown in Table 1(b). According to this, excluding

$\mathcal{T}$	A	B	C	D	E	F	G	H	I	J	K	L
A	0.0	0.0	$\frac{1}{2}$	$\frac{3}{5}$	0.0	$\frac{2}{3}$	0.0	0.0	0.0	$\frac{1}{2}$	0.0	0.0
B	0.0	0.0	$\frac{1}{2}$	0.0	$\frac{3}{5}$	0.0	0.0	$\frac{2}{3}$	0.0	0.0	0.0	0.0
C	$\frac{3}{8}$	$\frac{3}{8}$	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
D	$\frac{2}{8}$	0.0	0.0	0.0	$\frac{2}{5}$	0.0	0.0	0.0	0.0	0.0	0.0	0.0
E	0.0	$\frac{2}{8}$	0.0	$\frac{2}{5}$	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
F	$\frac{3}{16}$	0.0	0.0	0.0	0.0	0.0	$\frac{1}{2}$	0.0	0.0	0.0	0.0	0.0
G	0.0	0.0	0.0	0.0	0.0	$\frac{1}{3}$	0.0	$\frac{1}{3}$	0.0	0.0	0.0	0.0
H	0.0	$\frac{3}{16}$	0.0	0.0	0.0	0.0	$\frac{1}{2}$	0.0	0.0	0.0	0.0	0.0
I	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	$\frac{1}{4}$	0.0	$\frac{1}{4}$
J	$\frac{3}{16}$	0.0	0.0	0.0	0.0	0.0	0.0	0.0	$\frac{1}{2}$	0.0	$\frac{1}{2}$	0.0
K	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	$\frac{1}{4}$	0.0	$\frac{1}{4}$
L	0.0	$\frac{3}{16}$	0.0	0.0	0.0	0.0	0.0	0.0	$\frac{1}{2}$	0.0	$\frac{1}{2}$	0.0

(a)

t	
A	0.183
B	0.183
C	0.137
D	0.076
E	0.076
F	0.051
G	0.034
H	0.051
I	0.034
J	0.068
K	0.034
L	0.068

(b)

Table 1. (a) $\mathcal{T}$ and (b) t for for the example Web graph in Figure 1

the vertices A and B themselves, the most dominant vertex is C. Vertices, D and E follow C with lower dominance values as they are on a longer path between A and B. Although vertices J and L, are on an even longer path, they follow D and E closely since they are on multiple paths. o

2.3 Case 2: $\mathcal{S}$ Contains More than Two Pages

In order to extend the algorithm presented in the previous section to the case in which $\mathcal{S}$ contains more than two seed pages, we need to observe that the algorithm uses these seed pages to discover the boundaries of the neighborhood, and to calculate the *penalty* of each vertex in the random walk graph.

The first of these tasks is easy to generalize. Given a set of vertices, $|\mathcal{S}| \geq 2$ and a radius, d, the relevant neighborhood, $G^N(V^N, E^N)$ of $G^u(V, E^u)$, is the set of vertices, $V^N = V_{G^u}(\mathcal{S}, d)$, that are reachable from the vertices in I in d edge traversals: $\left(\forall v_i \in V_{G^u}(\mathcal{S}, d) \quad \bigvee_{v_j \in \mathcal{S}} reachable_{G^u}(v_j, v_i, d)\right)$.

The second task, determining the penalty each vertex, can be handled in two ways. We can either trivially generalize the definition of the penalty as $\left(penalty(v'_i) = \sum_{v'_j \in \mathcal{S}} sdist(v'_i, v'_j)\right)$ or we can use

$$length(minimum\_steiner\_tree(\mathcal{S} \cup \{v'_i\}))$$

to get a more accurate picture of the distance of v'_i from the seed vertices. Note that the problem of finding the minimum weighted connected subgraph, G', of a given graph G, such that G' includes all vertices in a given subset R of G is known as the *Steiner tree* problem[1] [5]. Unfortunately, the minimum weight Steiner tree problem [6] is known to be NP-hard; i.e., it is not known whether there exists a polynomial time solution. The first option, on the other hand, is known to require polynomial time; and consequently, it is more efficient.

[1] If it exists, G' is guaranteed to be a tree.

2.4 Complexity of the Algorithm

For a given graph $G^N(V^N, E^N)$ the maximum degree of vertices is m, where $0 \leq m \leq |V^N|$, we analyze the complexity of the algorithm as follows:

1. The algorithm firsts find the shortest distance between every vertex and the seed vertices in $\mathcal{S}$. This operation can be performed using the Floyd-Warshall all pairs shortest path algorithm in $O(|V^N|^3)$. The assignment of penalties for each vertex, then, takes $O(|V^N|)$ time.
2. Once the penalties are know, calculation of the transition values for a given vertex takes $O(\mathcal{L}(m, m+1))$ time, where $\mathcal{L}(x, y)$ is the time that is required to solve a set of linear equations with x number of variables and y number of equations. Hence, the total number of time required to find all transition values in the graph is $O(|V^N| \times \mathcal{L}(m, m+1))$.
3. After all the transition values are known, the algorithm solves a set of linear equations with $|V^N|$ variables and $|V^N|+1$ equations. Hence, this step takes $O(\mathcal{L}(|V^N|, |V^N|+1))$.
4. Consequently, the total amount of time taken by the algorithm is

$$O(|V^N|^3 + |V^N| \times \mathcal{L}(m, m+1) + \mathcal{L}(|V^N|, |V^N|+1)).$$

3 Experiments

Our current implementation utilizes a linear equation solver `Maple` for calculating both edge transition probabilities and the corresponding convergence vector. We have conducted our experiments on *www-db.standford.edu*, which has 3600 pages and 12,581 edges. The average number of edges per page is 3.5. The experiments were ran on a 500MHz Pentium Architecture Linux OS PC with 128 MB of RAM.

3.1 Execution time

The first experiment is for measuring the execution time of the algorithm. Note that there are two phases: calculation of **(1)** the edge transition probabilities and of **(2)**the convergence vector. We measure their execution time separately.

For a Web subgraph containing neighborhood of 1085 nodes (i.e. pages within 3 links from seed URLs `/classes.html` and `~echang/`, the execution time is reasonable fast. The total clock time needed is 760 seconds for the first phase; among that 572.76 seconds (i.e. 75.36%) is for reading the Web graph from disk. And, the clock time needed for the second phase is 343 second, among that only 9 seconds are for writing the results.

Thus, the total CPU time needed is 187.24 seconds for the first phase to solve 1,085 sets of equations (one for each node with average 4 to 5 variables and 4 to 5 constraints). In the second phase, only one set of equations needs to be solve. This set of equations with 1,085 variables and 1,086 constraints requires 334.36 seconds to solved. We believe such execution time is satisfactory and can be further improved by using a faster machine with larger memory or passing the results of the first phase directly as the input of the second phase.

0.217939 /lore/	0.080059 /midas/midas.html
0.071164 /projects.html	0.056338 /c3/c3.html
0.053373 /~hector/infolab.html	0.053373 /projects-noim.html
0.053373 /projects-noim.html	0.048925 /tsimmis/tsimmis.html
0.040030 /tsimmis/	0.035582 /~chaw/

Table 2. Top 10 association pages within radius of 1 from seeds

0.066801 /lore/	0.015541 /tsimmis/tsimmis.html
0.049419 /www-db.stanford.edu/	0.015337 /~widom/
0.040217 /projects.html	0.014814 /c3/
0.038854 /~hector/infolab.html	0.014519 /people/gio.html
0.032719 /tsimmis/tsimmis.html	0.014315 /LIC/
0.028629 /projects-noim.html	0.013360 /~chaw/
0.028629 /projects-noim.html	0.013088 /~sergey/
0.028356 /~ullman/pub/hits.html	0.012270 /CHAIMS/
0.025630 /tsimmis/	0.012065 /people/hector.html
0.025175 /c3/c3.html	0.011043 /people/jpgs/
0.024539 /midas/midas.html	0.010225 /SKC/
0.022085 /people/widom.html	0.009979 /people/index.html
0.021813 /~widom/widom.html	0.009611 /people/index2.html
0.017109 /warehousing/warehouse.html	0.009161 /~ullman/
0.015678 /people/sergey.html	0.008930 /~tlahiri/

Table 3. 30 association pages within radius of 2 from seeds

3.2 Association Mining Results

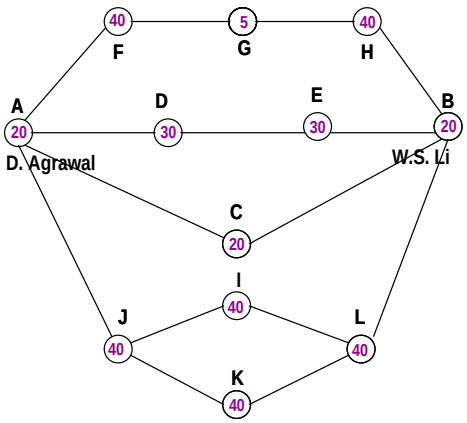
The second experiment is for testing the effectiveness of the algorithm. We can present only a small portion of the results due to the space limit.

For the project home page URLs `/lore/` and `/midas/midas.html`, the top 10 of 32 association pages within radius of 1 from seeds is shown in Table 2. We are satisfied with the results given most people working for both projects, such as `/~widom`, `/~ullman`, `/~wiener/`, and `/~sergey/`, as well as the home pages of related projects are selected. When we extend the exploration radius from 1 to 2, the results are still satisfactory given most of the results remain and few new pages are introduced. The associating pages within radius of 2 from seed URLs are sown in Table 3 (top 30 out of 155 pages are shown).

We observe that when the radius is extended, many index pages, such as the root page, departmental page (e.g. `infolab.html`, and more project pages (e.g. /warehousing/warehouse.html and /CHAIMS/), are now included. We also observe several other interesting factors. For example, Professor Wiederhold, whose home page is `/people/gio.html`, does not participate in `LORE` project and `MIDAS` project. However, Professor Wiederhold is the organizer of a popular database seminar where most people has links pointing to the seminar announcement page. As a result, the home page of Professor Wiederhold is selected.

4 Extension to Content-Focused Algorithm

In order to incorporate document contents to the association mining process, we propose to change the definition of penalty to also include document contents.



(a)

t	
A	0.123
B	0.123
C	0.092
D	0.051
E	0.051
F	0.116
G	0.185
H	0.116
I	0.023
J	0.046
K	0.023
L	0.046

(b)

Fig. 5. (a) Penalty of an example Web sub-graph, and (b) corresponding t

This variation, the *Content-Focused Random Walk Algorithm* , allows us to mine document associations with respect to not only seed URLs but also a particular topic. For example, we may ask a more specific question: "find why the pages `W.Li` and `D.Agrawal` are associated with respect to NEC" or "find why the pages `W.Li` and `D.Agrawal` are associated with respect to the `Y.Wu` page". Assuming that there exists a function, $relevance(v, topic)$, for measuring the relevance between the contents of a vertex v and a given topic *topic*, we redefine the penalty of a vertex as $\left(\frac{penalty(v)}{relevance(v,topic)}\right)$.

Alternatively, we can also consider content similarity of two linked pages to adjust the *importance* of each link. Intuitively, if a page is *content-wise* more related to the seed pages, then it is more likely to explain the association between them; hence it should be assigned a lower penalty, increasing its likelihood of being visited during a random walk. We call this variation `Content-Sensitive Random Walk Algorithm`. Assuming that there exists a function, $relevance(v, \mathcal{S})$, which evaluates the relevance of the content of a given vertex $v \in V$ with respect to a set of vertices $\mathcal{S} \in 2^V$, we can redefine the penalty of a vertex as $\left(\frac{penalty(v)}{relevance(v,\mathcal{S})+1}\right)$, $\left(\frac{penalty(v)}{relevance(v,\mathcal{S})}\right)$, or as $(penalty(v) \times (2 - relevance(v, \mathcal{S})))$. Note that the choice of the adjusted penalty function is application dependent and such a choice allows users to fine tune the sensitivity to the contents.

Example 4. Let us look reconsider the previous example. Now we want to ask "find why the pages `W.Li` and `D.Agrawal` are associated with respect to `Peter Scheuermann`. Let us assume that the page G has a strong relevance to the focused content, "Peter Scheuermann". Let us also assume that the relevance function used assigns 0.8 to G and 0.1 to all other pages.

Assuming that we use the penalty function $\left(\frac{penalty(v)}{relevance(v,I)}\right)$, Figure 5 shows the graph and the corresponding convergence matrix, t. According to this vector, the most "dominant" vertex in the graph is G. Comparing with the results in

Table 1(b), the scores of G, F, and H are boosted since G is now in focus. In this example, we observe that the results successfully reflect the structure and the document contents with respect to the given topic. o

5 Related work

Link information has been used by many search engines to rank query results. They assume that the quality of a document can be "assured" by the number of links pointing to it. HITS algorithm was proposed by J. Kleinberg [3]. It aims at selecting a small subset of the most "authoritative" pages from a much larger set of query result pages. Authoritative page is a page with many incoming links and a hub page is a page with many outgoing links. Such authoritative pages and hub pages are mutually reinforced: good authoritative pages are linked by a large number of good hub pages and vice versa. This technique organizes topic spaces as a smaller set of hub and authoritative pages and it provides an effective mean for summarizing query results, so called "topic distillation".

Bharat and Henzinger [7] improved the basic HITS algorithm [8] by adding additional heuristics. The modified topic distillation algorithm considers only those pages that are in different domains with similar contents for mutual authority/hub reinforcement. Another variation of the basic topic distillation algorithm is proposed by Page and Brin[9]. Their algorithm further considers page fanout in propagating scores.

Many of these basic and modified topic distillation algorithms have been also used to identify latent Web communities[10, 11]. These above techniques focus on finding high quality documents induced by link analysis. Our proposed algorithm can be extended for topic distillation by targeting all nodes in the explored graph instead of few seed URLs. By such adjustment, the algorithm would be able to find hub and authority.

Dean and Henzinger[1] proposed two algorithms, *companion* and *cocitation* to identify related pages and compared their algorithms with the Netscape algorithm[2] used to implement the `What's Related?` functionalities. By extending the scope from documents to Web sites, Bharat and Broder[12] conducted a study to compare several algorithms for identifying mirrored hosts on the Web. The algorithms operate on the basis of URL strings and linkage data: the type of information easily available from web proxies and crawlers.

This work above focuses on finding related documents or Web sites. Our work focuses on finding pages inducing associations of given seed URLs. The proposed random walks algorithm can be extended to be content-focused.

6 Concluding remarks

In this paper, we present a framework and an algorithm for mining implicit associations among Web documents induced by link structures and document contents. The algorithm works on any graph and can focus on a specific topic. We have implemented and evaluated the algorithm. The preliminary experimental results on real Web data show that the algorithm work well and efficiently.

The authors would like to express their appreciations to `www-db.stanford.edu` for its data used in their experiments. Selecting this Web site is due to the considerations (1) the authors need to be familiar with the contents so that the authors can evaluate the results; and (2) the pages in the Web sites must not be dynamically generated pages. The second consideration restricts the authors from using most of corporation sites. The experimental results presented in this paper are for the purposes of scientific research only.

References

1. Jeffrey Dean and Monika Henzinger. Finding Related Pages in the World Wide Web. In *Proceedings of the 8th World-Wide Web Conference*, Toronto, Canada, May 1999.
2. Netscape Communications Corporation. What's Related web page. *Information available at http://home.netscape.com/netscapes/related/faq.html.*
3. Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, pages 668–677, January 1998.
4. Wen-Syan Li and Selcuk Candan. Integrating Content Search with Structure Analysis for Hypermedia Retrieval and Management. To appear in *ACM Computing Survey*, 2000.
5. Frank K. Hwang, Dana S. Richards, and Pawel Winter, editors. *The Steiner Tree Problem (Annals of Discrete Mathematics, Vol 53).* 1992.
6. S.L. Hakimi. Steiner's problem in graphs and its implications. *Networks*, 1:113–131, 1971.
7. Krishna Bharat and Monika Henzinger. Improved algorithms for topic distillation in a hyperlinked environment. In *Proceedings of the 21th Annual International ACM SIGIR Conference*, pages 104–111, Melbourne, Australia, August 1998.
8. Soumen Chakrabarti, Byron Dom, Prabhakar Raghavan, Sridhar Rajagopalan, David Gibson, and Jon Kleinberg. Automatic Resource Compilation by Analyzing Hyperlink Structure and Associated Text. In *Proceedings of the 7th World-Wide Web Conference*, pages 65–74, Brisbane, Queensland, Australia, April 1998.
9. Lawrence Page and Sergey Brin. The Anatomy of a Large-Scale Hypertextual Web Search Engine. In *Proceedings of the 7th World-Wide Web Conference*, Brisbane, Queensland, Australia, April 1998.
10. David Gibson, Jon M. Kleinberg, and Prabhakar Raghavan. Inferring Web Communities from Link Topology. In *Proceedings of the 1998 ACM Hypertext Conference*, pages 225–234, Pittsburgh, PA, USA, June 1998.
11. Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, and Andrew Tomkins. Trawling the Web for Emerging Cyber-Communities. In *Proceedings of the 8th World-Wide Web Conference*, Toronto, Canada, May 1999.
12. Krishna Bharat and Andrei Z. Broder. Mirror, Mirror, on the Web: A Study of Host Pairs with Replicated Content. In *Proceedings of the 8th World-Wide Web Conference*, Toronto, Canada, May 1999.