Reasoning for Web Document Associations and Its Applications in Site Map Construction

K. Selçuk Candan* Wen-Syan Li

C&C Research Laboratories, NEC USA, Inc.
110 Rio Robles M/S SJ100, San Jose, CA 95134, USA
Email: {candan,wen}@ccrl.sj.nec.com
Tel:(408)943-3008 Fax:(408)943-3099

Abstract

Recently, there is an interest in using associations between web pages in providing users with pages relevant to what they are currently viewing. We believe that, to enable intelligent decisions, we need to answer the question "for a given set of pages, find out *why* they are associated". We present a framework for reasoning about Web document associations. We start from the observation that the reasons of the web page associations are implicitly embedded in the content of the pages as well as the links connecting them. The association reasoning scheme we propose is based on a *random walk algorithm*. This algorithm can take both link structure and contents into consideration and allows users to specify a focus. We then show how the proposed algorithm, combined with a logical domain identification technique, can be used for web site summarization and web site map construction to help users navigate through complex corporate sites. We see that, to achieve this goal, it is essential to recover the web authors' intentions and superimpose it with the users' retrieval contexts in summarizing web sites. Therefore, we present a framework, which uses logical neighborhoods, entry pages, and associations of entry pages, in creating context-sensitive summaries and maps of complex web sites.

Keywords. Reasoning about associations, link analysis, random walk, WWW, topic distillation, connectivity.

1 Introduction

In traditional information retrieval field, in order to determine the association between a given set of documents, keyword vectors that represents the contents of these document are compared. As the World Wide Web gets popular, hypertext and hypermedia is being employed in a wide range of information systems, and is a major means for guiding users while they access and navigate through information from distributed

*This work was performed when the author visited NEC, CCRL. He is currently an assistant professor at Arizona State University, Computer Science and Engineering Department.

sources. Hypermedia has two aspects: content and structural information. When an author prepares a web document, he/she would put intended information on the page while linking related information together using anchors. Thus, web structures can be used as clue to derive document association in addition to document contents. Recently, various techniques have been proposed to find the web document association, such as 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.

These document associations are used for providing users with pages relevant to what they are currently viewing. We, on the other hand, believe that in order to enable the user to make intelligent decisions, we also need to answer the question "for a given set of pages, find out *why* they are associated". In this paper, we present a framework for not only deriving document associations, but also inducing the reasons why they are associated. The techniques we propose go beyond the co-citation and social filtering approaches. We start from the observation that the reasons of the web page associations are implicitly embedded in the links connecting these pages. We now use an example to illustrate the problem statement and overview of our approach.

Example 1.1 In Figure 1, we show a set of links between two web pages `W.Li` and `D.Agrawal` (both are highlighted in the figure). The purpose of each web link is indicated as its label in the figure. For example, `W.Li` graduated from Northwestern University; whereas `D.Agrawal` and `P.Scheuermann` both graduated from SUNY. Based on this link structure, if we want to reason about the association between `W.Li`'s and `D.Agrawal`'s home pages, the link structure connecting `W.Li` and `D.Agrawal` are useful clues for discovering the underlying relationship embedded in the links. Below, we enumerate some of the reasons implicit in the link structure[1] that may explain why the home pages of `W.Li` and `D.Agrawal` are associated:

- *Reason 1:* `Web8 paper` page appears in a path of distance of 2 connecting the pages `W.Li` and `D.Agrawal`. Therefore, `W.Li` and `D.Agrawal` may be associated due to a co-authored paper.
- *Reason 2:* `Y.Wu` page is on two paths related to NEC Research Laboratories, each of distance 4. `W.Li` and `D.Agrawal` may be associated due to the fact they both supervised `Y.Wu` at different occasions or they participate in the same project at NEC.
- *Reason 3:* `WOWS'99` and `ACM DL'99` pages appear on a single path of distance 3. Such an association can be interpreted as that `W.Li` and `D.Agrawal` are participating in the same conference (e.g. presentation or program committee members).

[1] To be accurate, we should use *discovery* for the relationships we find and use the term *reasons* for the relationship once we validate it. In this paper, we use these two terms interchangeably.

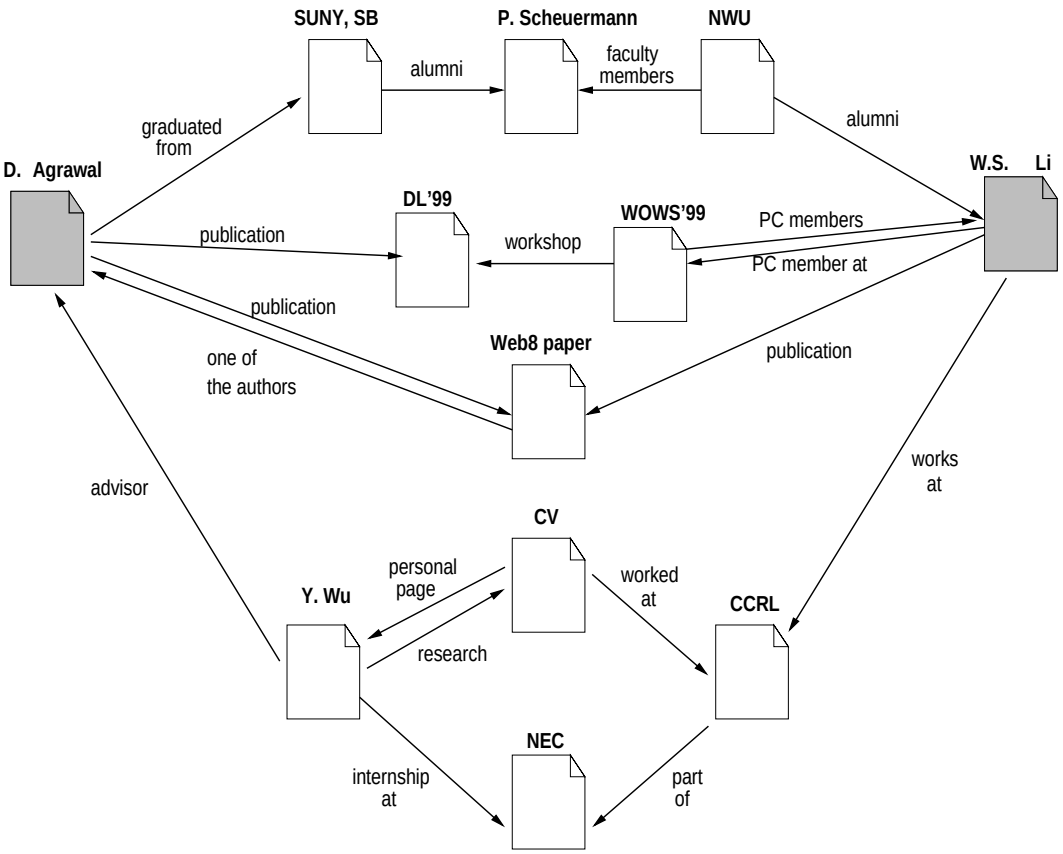


Figure 1: Link structure associating the web documents `W.Li` and `D.Agrawal`

- *Reason 4:* `P.Scheuermann`, `D.Agrawal`, `NWU`, `W.Li`, and `SUNY,SB` pages appear on a single path of distance 4. Such a link relationship implies that `D.Agrawal` and `W.Li` may be associated due to some people related to `SUNY, SB` they both know or due to an alumni relationship. □

The above example shows that the following two intuitions, along with the actual content of the pages, can generally be used to identify why a given set of pages are associated:

1. *A page that represent the association between input pages must be near them:* Pages on a shorter path between the `W.Li` and `D.Agrawal` pages are stronger indicators than others to reflect why the `W.Li` and `D.Agrawal` pages are associated.

2. *A structural representative must be highly connected to the given pages:* Pages which appear on more paths should be stronger indicators than others to reflect why the `W.Li` and `D.Agrawal` 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, a page with a high connectivity but far away from the seed URLs may be less significant for reasoning about the associations than a page with low connectivity but close to the seed URLs. A page which satisfies both of the above criteria (i.e. near seed URLs and with high connectivity) would be a good representative for the association.

Obviously, the *distance* between two pages can be defined in various ways. In the simplest case, the number of links between two pages can be used as the distance metric. On the other hand, in order to capture the physical as well as logical distances between pages, we will utilize different distance metrics capable of capturing document contents of user interests. For example, if want to reason the associations between `W.Li` and `D.Agrawal` with respect to NEC, then we may need to focus the reasoning on NEC.

Based on this motivation, we present a novel framework for reasoning associations among web documents using information *implicitly* embedded in the links connecting them, as well as contents of these connecting documents. Our reasoning scheme is based on a *random walk algorithm*. The algorithm has three major advantages. First, the algorithm can take both link structure and contents into consideration. Second, link analysis is performed by solving certain equations rather than by using iteration-based methods as in other approaches. Third, the approach allows users to specify a set of conditions/pages to focus the reasoning. In this paper, we validate the effectiveness of the algorithm on both syntactical and real web data.

The rest of the paper is organized as follows. In Section 2, we present the proposed framework and *random walk* algorithm for reasoning web document association using link structures. We discuss the algorithm in consideration of focused topics In Section 4 we describe how to adapt the proposed algorithm for constructing meaningful summarizations of web neighborhoods, which can be viewed and used as a web site map. In Section 5 we compare and contrast our approach with existing work. Finally we conclude this paper with discussion of future work.

2 *Random Walks* Algorithm for Reasoning Document Associations

In this section, we describe the scope of the proposed framework. First we describe the formal model we use for the association reasoning problem.

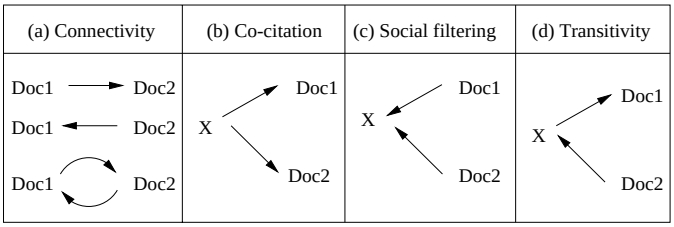


Figure 2: Associating documents by link

2.1 Formal Model

Let us assume that we are interested in reasoning about the associations of a set, $\mathcal{S} = \{s_1, \ldots, s_n\}$, of seed web pages (or $snodes$). The reasoning 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 reasoning. The cases where $\mathcal{S}$ contains more than two pages is discussed in Section 2.4.

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. It can depend on the computation resource or the judgment as to whether or not it is meaningful to consider the documents which are more than certain number of links away. 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 as shown in Figure 2. For example, Co-citation suggests that two documents are relevant to each other if they are linked via a common document and social filtering indicates that two documents are relevant to each other if they link to a common document. As we show in Figure 2, all types of link relationships are presented in the real world web environment and it is hard to judge what type of relationship is more important others. Thus, in this paper, we do not differentiate these four link relationships (*note that this generalization is not essential for the formalism or the algorithm and it can be easily extended for dealing with directions of the links*). Consequently, we can simplify the discussion by using only an undirected web graph, $G^u(V, E^u)$. Furthermore, assuming that we are given a radius d, we can 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 if there is no vertex $v_i \in V^N$, that is reachable from both v_a and v_b, then the radius d is not

large enough to contain an $inode$. Hence, without loss of generality, we will assume that there is at least one such vertex and consequently, the neighborhood graph, G^N, is connected.

To derive metrics for $Inode$ selection, one intuitive 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 (undirected simple) paths between the seeds, A and B, that pass through a candidate $inode$, v, and $length(p)$ is the length of the path p. Note that, although it merges the two required structural criteria, this metric has two main disadvantages preventing its use in reasoning association.

First, its calculation may require the enumeration of all paths in the graph, which may (in the worst case) require exponential time with respect to the size of the graph, and although the maximum length of the paths grows linearly with the number of vertices in the graph, the number of paths grows exponentially as shown in the experimental results on real web data shown in Appendix A. 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 in the graph.

Second, a score function working for one web graph may not work well for others since the scoring function is really web graph dependent. Thus, adhoc functions are not proper for this work.

2.2 Random Walks as an Implicit Metric

Consequently, instead of explicitly defining a metric, we will use random walk techniques and choose a set of random walk parameters that will implicitly capture the essence of these observations.

Intuitively, a random walk graph is a graph where each edge has an associated *transition probability*. If we are visiting node n at time t, then when we move to another node, we can choose among one of the outgoing edges of n according to the associated edge probabilities. In a sense, a random walk starts at some node n_0, chooses a neighbor based on the transition probabilities, moves to the new neighbor, and repeats the process. If we continue this process L number of steps, then we have a probability distribution of which node the random walk might be at.

Stationary distribution of a random walk is a *convergence vector* (probability distribution), where no subsequent steps change the probability distribution. It can be shown that if an undirected random walk graph is strongly connected and non-bipartite, then it can be modeled as a Markov Chain with a stationary

distribution (the fundamental theorem of Markov Chains). If the graph is bipartite, then it can be converted into a non-bipartite graph by introducing self loops. Therefore, in this paper, without any loss of generality, we assume that the graph is bipartite.

Given a random walk graph $G(\mathcal{V}, \mathcal{E}, \mathcal{T})$, where T is a square matrix that denotes the edge transition probabilities, the convergence vector t can be found by solving the equation $t = \mathcal{T}t$ (i.e., the distribution is stationary, subsequent steps do not affect t) or, in other words, $(I - \mathcal{T})t = 0$. Note that if the matrix T is regular (i.e., determinant of T is non-zero), then t will have one and only one solution.

Intuitively, the convergence probability of a node n gives the ratio of the time spent at that node in an sufficiently long random walk. Therefore, it captures the connectivity of the graph well. In fact, random walks have been used by various researchers to identify the link structure. Google [5] introduces a PageRank algorithm that performs a random walk on the WWW independent of the query. SALSA uses random walks to identify hubs and authorities in a web graph [6, 7].

However, neither of these approaches are suitable for site map construction as they only consider connectivity aspect of the problem; i.e., they ignore seed nodes. Our goal on the other hand, is to use seed nodes and identify the nodes in the graph that are not only well connected, but also close to the seeds, and also have content relevant to the site map context. In this section, we show how to achieve this goal by appropriate selection of transition probabilities.

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

In order to materialize this observation into an algorithm, we define and construct a random walk graph that reflects the required random walk parameters.

Definition 2.1 (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 $\sum_{1 \leq j \leq |\mathcal{V}|} \mathcal{T}[j, i] = 1.0$ $\diamond$

Algorithm 2.1 (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 3. $\diamond$

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:

$$L(v'_i) = \left\{ \sum_{\langle v'_i, v'_j \rangle \in \mathcal{E}} \mathcal{T}[j,i] = 1.0 \right\} \bigcup \left\{ \mathcal{T}[j,i] \times penalty(v'_j) = \mathcal{T}[k,i] \times penalty(v'_k) \;\middle|\; \langle v'_i, v'_j \rangle \in \mathcal{E} \text{ and } \langle v'_i, v'_k \rangle \in \mathcal{E} \right\}$$

Figure 3: Algorithm for constructing a random walk graph

2.3.1 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, the probability that the random walk process will leave vertex v_i (that is, it will proceed to one of its neighbors) must be

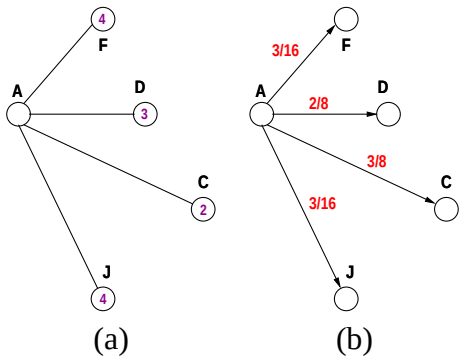


Figure 4: (a) Penalty of each node, and (b) transition values of each node

equal to 1.0.

Below, we first provide an example which shows the effect of the penalty term on the transition probabilities and which demonstrates the workings of Step 5 in greater detail.

Example 2.1 Let us reconsider our running example and let us focus on the portion of the graph shown in Figure 4(a), which depicts the vertex A, its four neighbors (vertices F, D, C, and J), and the associated penalties (4, 3, 2, and 4 respectively) calculated according to the distance metric (for the sake of simplicity, Figure 4(a) shows only the the five vertices relevant to this example and omits the phase of penalty calculation).

The following items reflect *some* of the facts we know about the transition probabilities of the edges that are leaving vertex 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.

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

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

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

Note that the above set of constraints is redundant (only first four equations are enough to solve for all the unknowns). Figure 4(b) shows the transition values obtained through the solution of these linear constraints. □

Once the edge transition probabilities that reflect the structural information is calculated, the convergence vector (or the eigenvector corresponding to the linear equations) can be used to identify the $inodes$.

Definition 2.2 (Convergence Vector) Given a random walk graph $\mathcal{R}_{(v_a, v_b, d)}(\mathcal{V}, \mathcal{E}, \mathcal{T})$, t is a convergence vector[2] of $\mathcal{T}$. That is $t = \mathcal{T}t$ or in other words $(I - \mathcal{T})t = 0$. ◇

c 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, a better* $inode$ *is the corresponding vertex. Consequently, we choose the* $inodes$ *using their corresponding values in the convergence vector.*

Definition 2.3 ($Inode$ **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. Note that the $inode$ vertex is the vertex that best *reflects* the association between vertices v_a and v_b. ◇

Next, we show how the algorithm works using our running example.

Example 2.2 Let us assume that Figure 5(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 5(a) are the corresponding distance penalties of the vertices. These penalties will be used in determining the dominant vertex of the graph.

[2] $\sum_{1 \leq i \leq |V'|} t[i] = 1.0$

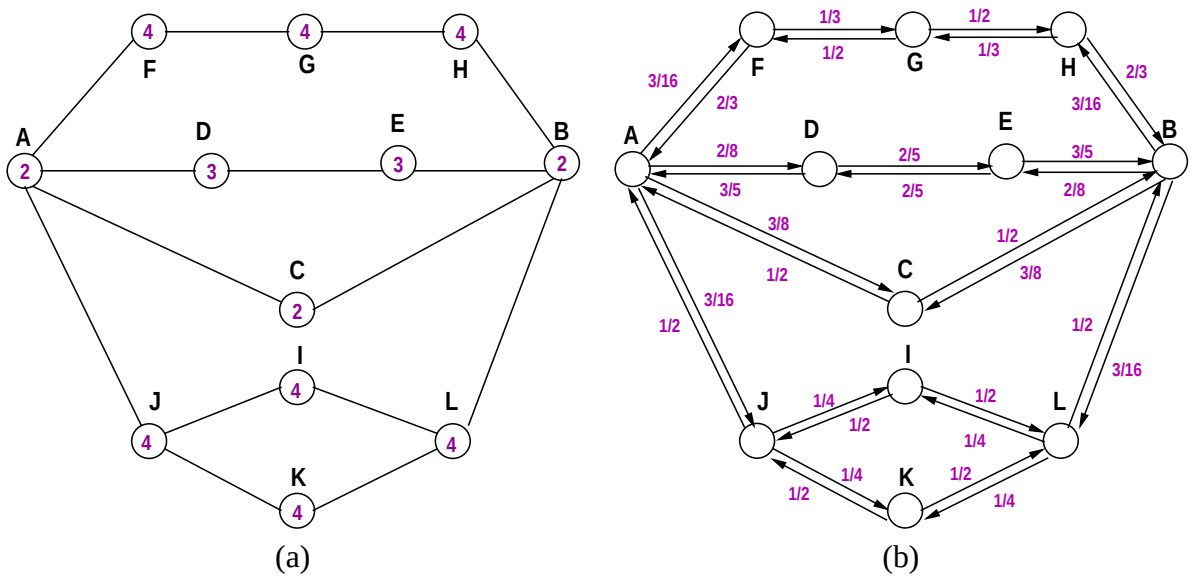


Figure 5: (a) Penalty values, and (b) transition values for the example web graph in Figure 1

Figure 5(b), then, shows the random walk graph, $\mathcal{R}_{(A,B,2)}$, that corresponds to the graph shown in Figure 5(a). The transition values that are calculated using the vertex penalties 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 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. □

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

Now, we are ready to extend the structural $inode$ selection to the cases where there are more than two input 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:

$$\forall v_i \in V_{G^u}(\mathcal{S}, d) \quad \bigvee_{v_j \in \mathcal{S}} reachable_{G^u}(v_j, v_i, d)$$

$\mathcal{T}$	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

Again, we assume that there is at least one vertex $v_i \in V_{G^u}(\mathcal{S}, d)$, that is reachable from all seed vertices.

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

$$penalty(v_i') = \sum_{v_j' \in \mathcal{S}} sdist(v_i', v_j'), \text{ or}$$

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

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[3] [8]. Consequently, an accurate definition of the penalty would be the later penalty definition. Unfortunately, the minimum weight Steiner tree problem [9] is known to be NP-hard; i.e., it is not known whether there exists a polynomial time solution. Having the penalty definition we used in the algorithm, which is calculated as the sum of all relevant shortest paths, is known to require polynomial time. Consequently, it is more efficient.

2.5 Case 3: Consideration of Focused Topics

The algorithm presented in previous two subsections was using the vertex-to-seed distance information to identify penalties used in transition probability calculations for the random walk. In order to incorporate document contents to the task of reasoning association, we propose to change the definition of penalty to also include document contents. Here we present an extension to the basic *random walk* algorithm by considering specific topic that users are interested in reasoning.

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

The variation of *Topic-Focused Random Walk Algorithm* allows us to reason 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 `P. Scheuermann` page". To answer these two questions, the pages related to NEC or the `P. Scheuermann` page would be assigned lower penalty values so that these pages would receive higher scores.

Assuming that there exists a function, $relevance(v, topic)$, for measuring the relevance between the contents of a vertex v and a given topic $topic$. The adjustment for the content-focused random walk algorithm is to redefine the penalty of a vertex as $\frac{penalty(v)}{relevance(v,topic)}$.

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.

Again, assuming that there exists a function, $relevance : V \times 2^V \rightarrow [0.0, 1.0]$, 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:

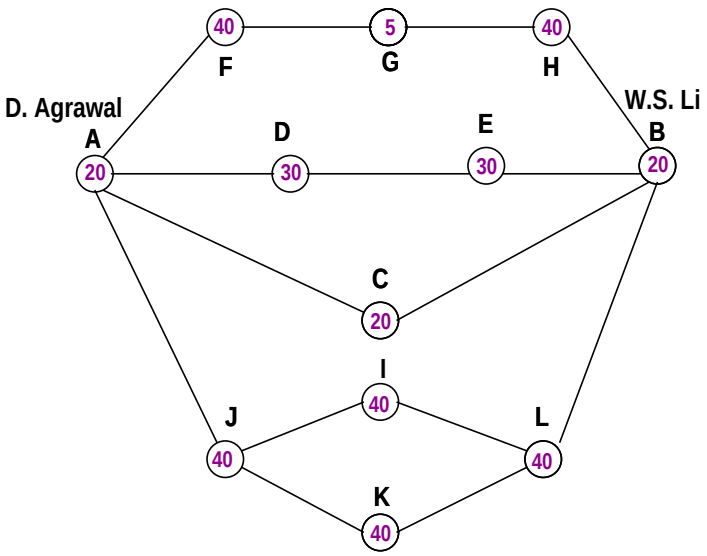
$$\begin{aligned} penalty\_with\_content_1(v) &= \frac{penalty(v)}{relevance(v, \mathcal{S}) + 1}, \\ penalty\_with\_content_2(v) &= \frac{penalty(v)}{relevance(v, \mathcal{S})}, \quad \text{or} \\ penalty\_with\_content_3(v) &= penalty(v) \times (2 - relevance(v, \mathcal{S})). \end{aligned}$$

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 2.3 Let us look at the graph used in the previous examples again. 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. Let us also assume that we use the following penalty function:

$$\frac{penalty(v)}{relevance(v, I)}.$$

Figure 6(a) shows the corresponding graph and the associated vertex penalties. After the random walk graph and the transition matrix, $\mathcal{T}$ is constructed (we omit the details here), we can see the convergence



(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)

Figure 6: (a) Penalty of an example web sub-graph, and (b) corresponding t

matrix, t, corresponding to this graph is as shown in Figure 6(b). 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 due to G is in focus. In this example, we observe that the results successfully reflect the consideration on both link structures and document content with respect to a certain focused topic. □

2.6 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 random walk 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 (polynomial) 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

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

2.7 Summary

In this section we present a random walk algorithm for reasoning web document associations. The algorithm has two phases The first phase is to calculate the edge transition probabilities, in which both connectivity and link path length factors are considered. The second phase is to calculate a convergence vector for the edge transition probabilities. In the case of considering focused topic, the relevancy of each document to the focused topic is used to adjust the edge transition probabilities in the first phase. In our current implementation, we utilize a linear equation solver called `Maple` for calculating both edge transition probabilities and the corresponding convergence vector. In the next section, we present our experimental results.

3 Experiments

We have conducted a set of experiments on *www-db.standford.edu*, which has 3040 pages and $12,581$ edges. The average number of edges per pages is 3.5. The experiments were ran on a 500MHz Pentium Architecture Linux OS PC with 128 MB of RAM. The `Maple` linear equation solver is used for calculating edge transition probabilities and the corresponding convergence vector.

3.1 Execution time

The first experiment is to measure the execution of the algorithm. Note that there are two phases required calculation by the linear equation solver, the calculation of edge transition probabilities as the first phase and calculation the corresponding convergence vector as the second phase. 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. And, the clock time needed is 343 second for the second phase, 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 equation needs to be solve. That is, a set of equation with 1,085 variables and 1,086 constraints. It requires 334.36 seconds to solved. We believe such execution time is satisfactory and it can be further improved by using faster machines or modifying the `Maple` Linear Equation Solver so that it would pass the results of the first phase as the input of the second phase.

3.2 Association Reasoning Results without Consideration of Focused Topic

The second experiment is to test the effectiveness of the algorithm. In this section, we present a small, but representative, portion of the results due to the space limits. We asked our system to reason the associations between the two project pages `/lore/` and `/midas/midas.html` in the Stanford domain. We changed the neighborhood radius to see the effect of increasing web space.

Case 1: When the 32 pages within radius of 1 from the two seed URLs are ranked *without using the content of the pages*according to their association dominance, we get the following list:

Score	URL	Score	URL
0.217939	/lore/	0.017791	/tsimmis/tsimmis.html
0.080059	/midas/midas.html	0.016679	/~widom/
0.071164	/projects.html	0.013343	/people/widom.html
0.056338	/c3/c3.html	0.011119	/dbseminar/Archive/SpringY98/stanford-apr3.html
0.053373	/~hector/infolab.html	0.011119	/~vassalos/cv.html
0.053373	/projects-noim1.html	0.010007	/lore/top.html
0.053373	/projects-noim2.html	0.010007	/lore/nav.html
0.048925	/tsimmis/tsimmis.html	0.010007	/lore/data.html
0.040030	/tsimmis/	0.010007	/lore/corner.html
0.035582	/~chaw/	0.006672	/~wiener/
0.029652	/c3/	0.006672	/~sergey/
0.026686	/~widom/widom.html	0.006672	/people/wiener.html
0.023351	/~ullman/pub/hits.html	0.006672	/midas/title.html
0.017791	/~widom/xml-whitepaper.html	0.006672	/midas/menu.html
0.017791	/ tlahiri/	0.006672	/people/sergey.html
0.017791	/classix/biblio.htm	0.006672	/lore/addendum.html

In spite of the fact that these are generated without using the content of any pages, or any context description, we are satisfied with the results: pages of most people working for both projects, such as `/~widom`, `/~ullman`, `/~wiener/`, and `/~sergey/`, as well as the home pages of related projects are selected and reasonably ranked.

Case 2: 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 in to the top 30. The associating pages within radius

of 2 from seed URLs are as follows (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. This is because of the fact that these pages have high connectivities, and the page content information is required to avoid them. We also observe several other interesting facts. 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 and ranked relatively highly. Note also that the MIDAS page does not rank as high as some other pages. This is because of the fact that the MIDAS page itself does not describe the association between `LORE` and `MIDAS` projects as well as some other pages.

Score	URL	Score	URL
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-noim1.html	0.013360	/~chaw/
0.028629	/projects-noim2.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/

We also observe from these two experiments that most of the home pages are selected over other non-home pages due to their high connectivity. This characteristics is consistent with the intuition of the topic distillation technique. However, our technique can rank these pages according to the distance from *s-nodes*.

Another observation is that there are a few pages which are ranked high in the case of radius being set to 1 but they are ranked much lower in the case of radius being set to 2. Examples include /~widom/xml-whitepaper.html and /lore/top.html. We examine the graph and find out these pages are leaf nodes. That is, the links connecting them are ”explored” in the case 1 but there is no new link discovered when the radius is increased. As a result, they are ranked much lower in the case 2 than in the case 1.

3.3 Association Reasoning Results with Consideration of Focused Topics

In the third experiment we tested the effectiveness of the algorithm when a context is described by a user and the page contents are used to focus the association reasoning process with respect to this context.

We again asked our system to identify the associations between the two project pages `/lore/` and `/midas/midas.html` in the Stanford domain. However, this time we provided various keywords as contexts to observe the effect of focused association discovery.

Below, we provide the results for various keywords and for a neighborhood defined with a radius 2 from seeds. The relevancy measurement is by issuing queries to HotBot[10] with the focused topic as the keywords. The relevancy scores are between 0.6 and 1. The original costs are adjusted by being divided by the relevancy measurement.

Case 1: The top 20 associations with keyword "Ullman".

Score	URL	Score	URL
0.061468	/lore/	0.022616	/midas/midas.html
0.059844	/~widom/widom.html	0.022268	/~sergey/
0.045231	/	0.021417	/c3/c3.html
0.037693	/~hector/infolab.html	0.019832	/people/widom.html
0.034213	/projects.html	0.017918	/warehousing/warehouse.html
0.030618	/tsimmis/tsimmis.html	0.017281	/~ullman/fcdb.html
0.027950	/~ullman/pub/hits.html	0.014091	/~widom/
0.024587	/tsimmis/	0.013337	/people/sergey.html
0.024355	/projects-noim1.html	0.013221	/tsimmis/tsimmis.html
0.024355	/projects-noim2.html	0.012603	/c3/

Note that the result of this focus was that Prof. Ullman's pages increased their position in the association ranking, whereas some unrelated pages are ranked lower.

Case 2: The top 20 associations with keyword "papers or publications". The result of the change in the focus was that publication heavy pages are maintained at the top 20 ranking. This is especially visible when these results are compared to the previous focus. Many home pages which do not have a publication focus has been pushed down, whereas the publication pages has been maintained in the top 20 associations. For example, the publication page of Prof. Ullman is now ranked higher than his home page.

Score	URL	Score	URL
0.079927	/tsimmis/tsimmis.html	0.020947	/~ullman/pub/hits.html
0.057918	/lore/	0.019306	/tsimmis/
0.046335	/~widom/widom.html	0.018984	/c3/c3.html
0.042763	/	0.017376	/midas/midas.html
0.039578	/~hector/infolab.html	0.017231	/warehousing/warehouse.html
0.033786	/projects.html	0.015928	/SKC/
0.031276	/people/widom.html	0.014769	/people/anand.html
0.025581	/projects-noim1.html	0.011584	/LIC/
0.025581	/projects-noim2.html	0.011294	/LIC/mediator.html
0.023457	/people/gio.html	0.011101	/people/sergey.html

Case 3: The top 20 associations with keyword "papers or publications", with a higher sensitivity on the context describing keywords.

Score	URL	Score	URL
0.120948	/tsimmis/tsimmis.html	0.028512	/
0.073733	/people/gio.html	0.025364	/publications.html
0.053529	/people/widom.html	0.025090	/people/wiener.html
0.045898	/SKC/	0.022043	/projects.html
0.040298	/~widom/widom.html	0.018419	/projects-noim1.html
0.039858	/lore/	0.018419	/projects-noim2.html
0.037358	/people/anand.html	0.015519	/warehousing/warehouse.html
0.036400	/LIC/mediator.html	0.012078	/~wiener/papers.htm
0.035631	/infolab.html	0.011410	/c3/c3.html
0.030525	/people/joachim.html	0.011346	/tsimmis/

The result of the increase in the keyword sensitivity is that the focus has been further directed to the paper and publication heavy associations. For instance, while the CS departments main page have been pushed down (despite of the high connectivity) several publication heavy pages, such as "/~wiener/papers.html" and "/publications.html" have risen.

4 Applications to Web Site Map Construction

Complex web sites continue to proliferate, as web based information infrastructure in corporate and e-commerce organizations become integral parts of their business operations. Yet, due to the continuously increasing size of these infrastructures, it is getting ever difficult for users to understand and navigate through such sites. This shows itself as an increasing need in effective Web site map construction techniques. Furthermore, these complex structures prevent search engines from providing satisfactory context-sensitive retrieval functionalities. We see that in order to overcome this obstacle, it is essential to use

techniques that recover the Web authors' intentions and superimpose it with the users' retrieval contexts in summarizing Web sites. Therefore, in this paper, we present a framework for using implicit associations among web documents to discover web-pages and link structures that a relevant for a user's navigational goals.

Many large Web sites, such as Geocities[11], AOL[12], and NEC BIGLOBE[13], are either ISP sites or Web site hosting providers. Many pages in these sites are actually "logical domains" by themselves; i.e., *they form groups, where each page in a group have a specific semantic relation and a syntactic structure that relates it to the others in the same group.* For example, a group of pages linked to each other in a Web site containing information specific for "a user home page", "a project group", or "a tutorial on XML" can be viewed as logical domains. Logical domains are helpful in partitioning a physical domain into multiple smaller logical domains, suitable for presented as a site map and summarization, after being appropriately filtered.

When an author or a web designer prepares a logical domain, he/she would put intended information not only on in the textual content of a page; but, also on the link structure of the web site. Thus, the content of web pages along with the structure of the web domain can be used to derive hints to summarize web sites. What differentiates our work from similar work in the literature is that, we propose to use *associations between documents in a neighborhood and the reasons why they are associated* in the summarization process. Knowing these reasons, among other things, is essential in (1) in creating web site maps that matches web designers' intentions and (2) in superimposing the logical structure of a web site with the context provided by a visitors interests.

In this section, we describe how we can use the association mining algorithm we discussed earlier for Web site map construction. Since the topic-focused association reasoning algorithm brings together the interest of the user (keywords) and the structure of the Web space imposed by the designers, the proposed algorithm is a suitable tool for constructing Web sites.

4.1 Overview of the Web Site Summarization Process

In this subsection, we introduce the web site summarization task and discuss how to use document associations for this purpose. We see that corporate sites, and most of the web space, is composed of two types of neighborhood: physical and logical. The physical neighborhoods are separated from each other through domain boundaries or directory names. The logical neighborhoods, on the other hand, are mostly overlapping and they can cover multiple domains or they can be limited to a part of a single domain. Web designers usually aim at overlapping the physical neighborhoods of the web site with the logical neighborhoods. However, as (1) the foci of users may differ from each other, (2) the focus of a single user may

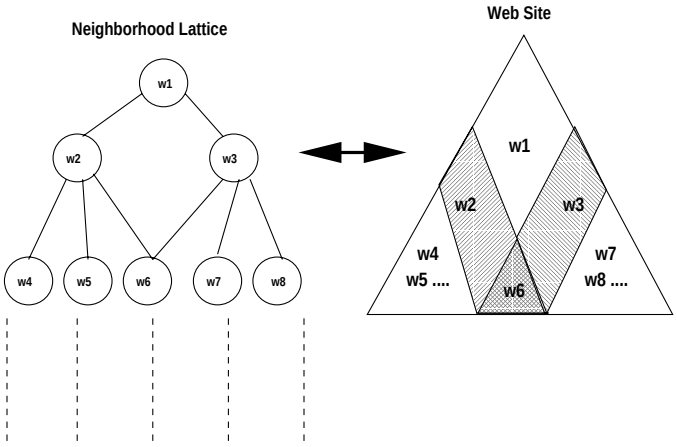


Figure 7: Logical neighborhoods of a web site

shift from time to time, such a strict design may loose its intended effect over time. Therefore, in order to create a dynamic site map, it is imperative to use both physical and logical neighborhoods of a corporate site simultaneously:

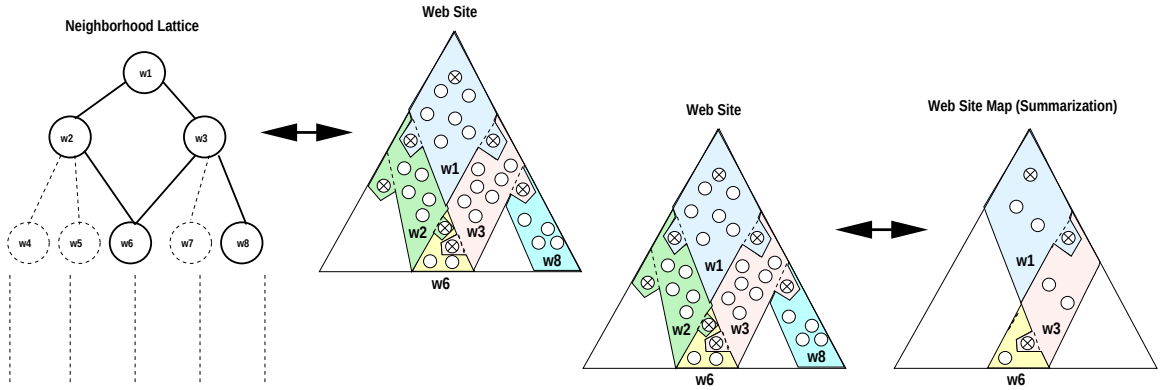
1. Physical neighborhoods are usually decided by the URL structures of the web sites. Discovering logical neighborhoods, however, require rules based on link structure, path information, document metadata, and citations to identify logical domain entry pages. These rules need to be automatically adjusted using a novel decision tree algorithm and training data provided by human feedback. We also develop techniques to define the boundary of each logical domain based on identified logical domain entry pages.

2. Once a corporate web site, W, is already into its logical neighborhoods, this structure can be modeled as a partially ordered lattice. $\mathcal{W}$:

   - each vertex w_i in $\mathcal{W}$ is a set of nodes (or a neighborhood) and
   - if w_j is a child of w_i, then w_j is a sub-neighborhood of w_i; furthermore, $w_i \cap w_j = \{v_{ij}\}$, where v_{ij} is the entry point to sub-neighborhood w_j from neighborhood w_i (Figure 8(a)).

   Intuitively, the lattice corresponds to a hierarchy of neighborhoods. At the highest level, we have a neighborhood which consists of high-level corporate pages and the entry pages of lower neighborhoods. Similarly, each neighborhood consists of a set of high-level pages and the entry-pages of all its sub-neighborhoods. Consequently, summarization of W involves of two tasks:

   (a) identification of important logical neighborhoods; or the nodes in the partially ordered lattice $\mathcal{W}$ that will be shown to the user (i.e., focusing on the neighborhoods) and



(a) Entry points (b) Summarization

Figure 8: The crossed circles denote the entry points of neighborhoods. Each sub-neighborhood contain one entry point per its parent neighborhoods. Each parent neighborhood includes the entry points of its sub-neighborhoods.

(b) summarization of each focused neighborhood, by identifying important nodes, based on user interest, web structure, and site content (Figure 8(b)).

In the next subsections, we discuss these to tasks in greater detail.

4.2 Identifying Logical Domains

Logical domains are identified based on the directory as well as link structures between pages. For instance, in user web sites, home pages are usually located at *user directories* which contain an entry page (e.g. index.html) that links to other user pages. Furthermore, these other pages tend to have 'back' or 'home' links to the original entry page. Based on these and other observations, we have developed a set of rules for identifying logical domain entry pages based on the available Web page metadata, such as title, URL string, anchor text, and link structures as well as popularity by citation [14]. Each rule has an associated scoring function. After all pages in a neighborhood are scored based on these rules, a set of pages with highest scores are chosen as entry page candidates to the logical domains. The boundaries of logical domains are then identified using path information and link structure. The details of the logical domain identification process are described in [14]. An example where 42 logical domain entry pages identified from 3040 pages in the Web site `www-db.stanford.edu` is shown in Figure 9.

4.3 Summarization of a Neighborhood

Once the logical neighborhoods are identified, (1) we need to identify those neighborhoods that are in focus and (2) we need to create a map which captures these logical neighborhoods and the link structures.

- http://www-db.stanford.edu
  - http://www-db.stanford.edu/cs347/
  - http://www-db.stanford.edu/people/
  - http://www-db.stanford.edu/pub/gio/
  - http://www-db.stanford.edu/tsimmis/
  - http://www-db.stanford.edu/warehousing/
  - http://www-db.stanford.edu/~abitebou/Abiteboul.html
  - http://www-db.stanford.edu/~chenli/
  - http://www-db.stanford.edu/~cho/
  - http://www-db.stanford.edu/~crespo/
    - http://www-db.stanford.edu/~crespo/publications/
      - http://www-db.stanford.edu/~crespo/publications/webwriter/
        - http://www-db.stanford.edu/~crespo/publications/webwriter/Overview.html
      - http://www-db.stanford.edu/~crespo/publications/awareness/pisa/
      - http://www-db.stanford.edu/~crespo/publications/meteor/wwii/
  - http://www-db.stanford.edu/~cyw/
  - http://www-db.stanford.edu/~danliw/
  - http://www-db.stanford.edu/~echang/
  - http://www-db.stanford.edu/~fangmin/
  - http://www-db.stanford.edu/~gio/
  - http://www-db.stanford.edu/~jan/watch/intro.html
  - http://www-db.stanford.edu/~junyang/seven/
  - http://www-db.stanford.edu/~sergey/
    - http://www-db.stanford.edu/~sergey/349/
  - http://www-db.stanford.edu/~testbed/python/manual.1.3/lib/
    - http://www-db.stanford.edu/~testbed/python/manual.1.3/lib/AllModules.html
    - http://www-db.stanford.edu/~testbed/python/manual.1.3/lib/Concept-Index.html
    - http://www-db.stanford.edu/~testbed/python/manual.1.3/lib/Function-Index.html
    - http://www-db.stanford.edu/~testbed/python/manual.1.3/lib/Module-Index.html
    - http://www-db.stanford.edu/~testbed/python/manual.1.3/lib/Top.html
    - http://www-db.stanford.edu/~testbed/python/manual.1.3/lib/Variable-Index.html
  - http://www-db.stanford.edu/~ullman/
    - http://www-db.stanford.edu/~ullman/fcdb.html
    - http://www-db.stanford.edu/~ullman/ullman-books.html
  - http://www-db.stanford.edu/~vassalos/
    - http://www-db.stanford.edu/~vassalos/defense/
  - http://www-db.stanford.edu/~wangz/
  - http://www-db.stanford.edu/~widom/
    - http://www-db.stanford.edu/~widom/widom.html
  - http://www-db.stanford.edu/~wilburt/
  - http://www-db.stanford.edu/~zhuge/
    - http://www-db.stanford.edu/~zhuge/zhuge.html

Figure 9: Logical domains identified in `www-db.stanford.edu`

4.3.1 Identification of Logical Neighborhoods in Focus

In order to identify those neighborhoods that are in focus, we need to start from the root neighborhood, w_1 of $\mathcal{W}$. This neighborhood contains, the high-level pages of the given corporate site along with the entry level pages of it sub-neighborhoods. Let us assume that we are interested in a predetermined number, k, of focus points in this top neighborhood. Then, we could rephrase the focus identification problem in terms of the neighborhood summarization problem:

- In order to identify the k focal points of the neighborhood w_1 of $\mathcal{W}$, summarize w_1 into a graph of size k. The k remaining pages are in user focus.
- Let us assume that $F = \{v_1, v_2, \ldots, v_k\}$ are the k pages in the summary of w_1. If $v_i \in F$ is an entry-page of a sub-neighborhood, w_i, then repeat the same process for the sub-neighborhood w_i.

Note that the above recursive process allow us to identify the focal points of a web site. The parameter k is user-dependent and describes how focused the user would like to be: smaller values of k correspond to more focused site maps. Note also that, this recursive algorithm assumes that given a neighborhood, we can identify k pages that should be in the summary. Next, we describe how we can achieve this task.

4.3.2 Identification of the Most Representative Pages in the Logical Neighborhood

Each neighborhood, $w_i \in \mathcal{W}$ consists of a set of neighborhood pages and the entry-pages of its sub-neighborhoods. Also, from each of its parent-neighborhoods, w_i is reached through one entry-page (Figure 8). Let us assume that the set, $\mathcal{E}$, of pages correspond to the entry pages of w_i from all its parent-neighborhoods that are in focus. Then our goal is to summarize the neighborhood with respect to these entry points as well as the content-description provided by the user. The summarized neighborhood will give the focused pages in this neighborhood and the corresponding connectivity.

In order to summarize a given neighborhood, we first have to identify the pages that are important. In this case, the entry pages of a neighborhood (from parents in focus) are relatively important as they will connect the web site maps of the neighborhoods. Note also that the entry pages of the sub-neighborhoods are also important as they will extent the map downwards in the hierarchy, given that the lower neighborhoods are also in focus.

Therefore, given a neighborhood, w_i, the set, $\mathcal{E}$, of focused entry pages from its parents, and the set, $\mathcal{L}$, of entry-pages to its sub-neighborhoods, we can create a set of seed pages (for summary) $\mathcal{S} = \mathcal{E} \cup \mathcal{L}$. Alternatively, if we would like to differentiate between vertices in $\mathcal{E}$ and $\mathcal{L}$, then we can introduce two new virtual vertices v_e and v_l such that v_e is connects to all vertices in $\mathcal{E}$ and v_l connected from all vertices in $\mathcal{L}$. We can then construct the set of seed vertices as $\mathcal{S} = \{v_e, v_l\}$. In either case, our goal is,

1. Let $G^N(V^N, E^N) \subseteq G^u$ be the neighborhood graph;
2. $V^\sigma = \emptyset$; $E^\sigma = \emptyset$;
3. Let $\mathcal{K} \subseteq V_{G^N}$ be the set of k vertices with the highest dominance values;
4. $V^\sigma = \mathcal{K}$;
5. For each v_i and $v_j \in V^\sigma$
   (a) $V_{temp} = V^N - \{V^\sigma - \{v_i, v_j\}\}$;
   (b) $E_{temp} = \{\langle v_k, v_l\rangle \mid (\langle v_k, v_l\rangle \in E^N) \wedge (v_k \in V_{temp}) \wedge (v_l \in V_{temp})\}$
   (c) If $sp(v_i, v_j)$ is the shortest path in $G_{temp}(V_{temp}, E_{temp})$ between v_i and v_j then
      i. Let the length of the path $sp(v_i, v_j)$ be Δ
      ii. $E^\sigma = E^\sigma \cup \{e = \langle v_i, v_j\rangle\}$;
      iii. If v_j is reachable from v_i in the directed graph G through the vertices in $sp(v_i, v_j)$, but if v_i is not reachable from v_j, then $\delta(e) = \langle \Delta, right\rangle$
      iv. If v_i is reachable from v_j in the directed graph G through the vertices in $sp(v_i, v_j)$, but v_j is not reachable from v_i, then $\delta(e) = \langle \Delta, left\rangle$
      v. If v_i and v_j are reachable from each other in the directed graph G through the vertices in $sp(v_i, v_j)$, then $\delta(e) = \langle \Delta, bi\rangle$
      vi. If neither v_i nor v_j is reachable from the other in the directed graph G through the vertices in $sp(v_i, v_j)$, then $\delta(e) = \langle \Delta, none\rangle$

Figure 10: Algorithm for constructing a summary

- given the set, $\mathcal{S}$, of seed (entry) web pages,
- potentially a content-description,
- a web neighborhood, $G^N = w_i$ which contains these seeds, and
- an integer k,

to create a *summary*, with k pages, of the neighborhood with respect to the seed pages.

Observation: Since, the web site map is a set of representative nodes in a web site, the nodes in a web site map needs to satisfy the following criteria:

- High connectivity so that users can navigate from these web site map nodes to other nodes easily.
- The contents of these web site map nodes need to be representative.

Thus, the nodes selected to form a web site map need to be both structural and content-wise representative for all pages in a web site. Therefore, we can use the selected nodes, *which describe the association between the pages in the site,* as a summary of a web site and to form a site map.

In Section 2, we described how to identify representative nodes using a random-walk based algorithm.

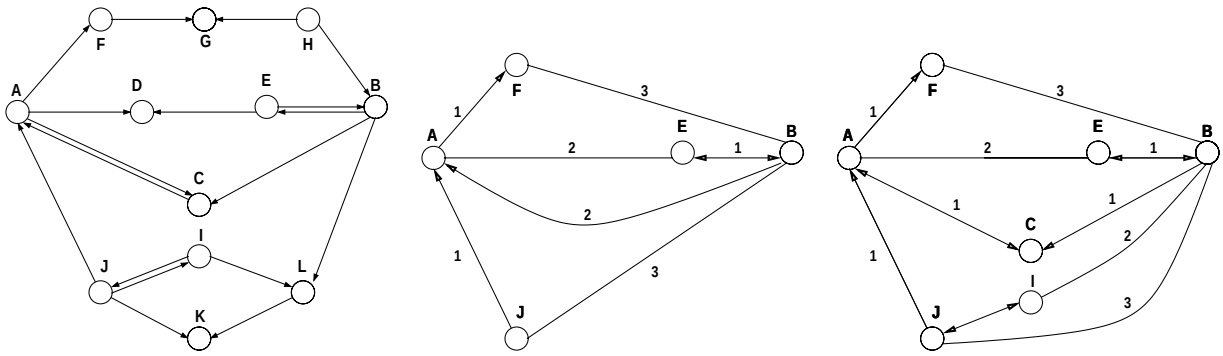
4.4 Creating a Summary Map Using the Dominant Pages

Given a graph $G(V, E)$, its undirected version $G^u(V, E^u)$, and the k *dominant vertices* in V with respect to the given seed vertices $\mathcal{S}$, we can construct a k-summary $\Sigma^k_{(\mathcal{S})}(V^\sigma, E^\sigma, \delta)$ of the input graph (δ is a mapping, $E^\sigma \rightarrow R^+ \times \{left, right, bi, none\}$, which describes the lengths and directions of the summarized edges) as shown in Figure 10.

Note that Step 3 of the algorithm requires the identification of the k most dominant vertices (or the vertices which describe the document associations the best) in the graph with respect to the seed vertices. These vertices will be the only vertices used in the summarization (Step 4). Given two dominant vertices, v_i and v_j, (to be visualized in the summary) Step 5 of the algorithm first constructs a temporary graph, $G_{temp}(V_{temp}, E_{temp})$, from the original web graph, such that no path between v_i and v_j can pass through another dominant node. Then, it uses the shortest path, $sp(v_i, v_j)$, between v_i and v_j in this temporary graph to identify edges that are to be visualized to the user. Consequently, a given edge that is included in the summary denote the shortest path, between two dominant vertices, that do not pass through other dominant vertices. Hence, this step eliminates the possibility of inclusion of redundant edges.

Note that, for the identification of shortest paths to be visualized, the path length can be defined in various ways. One possibility would be *minimization of the number of edges on the path.* This would be useful when the aim is to display the user information about the connectivity of the summarized graph. Alternative approaches include, *minimizing the sum of the visualization penalties of the vertices along the path*, to visualize the paths between the vertices in the summary graph that go through dominant vertices; and *maximizing the product of the transition probabilities on the edges of the path*, to highlight paths that are more likely to be followed according to the transition probabilities. Once the edges to include in the summary are known, the sub-steps of Step 5(c) gathers more information regarding each edge (such as the reachability of the two pages at the end-points from each other in the inherently directed web) and reflects this information to the summary in the form of edge labels.

Example 4.1 *Let us consider the graph in Figure 11(a). and let us assume that we are asked to construct* 5*- and* 7*-summaries of it. For this example, let us also assume that after running an association mining algorithm, we have found that the seven dominant vertices in this neighborhood (relative to a set of seed vertices) are* A*,* B*,* F*,* E*,* J*,* C*, and* I*. For simplicity, let us also assume that the length of the path is defined using the number of edges on it. Figures 11(b) and (c) shows the corresponding* 5*- and* 7*-summaries of this graph. The labels of the edges denote the length of the corresponding paths and the arrows on the edges denote the reachability of the end-points from each other (e.g., an edge of the form "*$\longleftrightarrow$*" denotes that both end-points can reach the other one over the web through this shortest path;*



(a) (b) (c)

Figure 11: (a) The web neighborhood in our running example, (b) its web site map with 5 nodes, and (c) its web site map with 7 nodes

whereas, an edge of the form "—" denotes that neither of the end-points can reach the other one over the web through this shortest path.

Those entry pages which are still in the map after the summarization are called *focused entry-pages*, and they point to the other logical domains that have to be further explored and summarized. Therefore, we recursively apply the summarization algorithm described above for those domains who have at least one *focused entry-page*.

4.5 Experimental Results

Using the same setup we used for the earlier experiments, we have conducted a set of experiments to validate the Web site summarization algorithm presented in this section. First of all, our system identified 42 logical domains among 3040 pages. These domains formed a tree-like structure as depicted in Figure 9. Then, we used the algorithm described in this section, to recursively summarize this logical structure with respect to the defined context.

The algorithm started by summarizing the root logical domain which consists of all the entry-pages of the logical domain in the second level and the all the page in root logical domain, where *www-db.stanford.edu* is the entry page. Thus, 1584 pages are included for the experiments. We insert a virtual node at the bottom and connected it to all leaf pages. The result of this summarization, using a radius of 2, is as follows (we omit the summarized edge to simplify the discussion):

Ranking	URL
0.124798	http://www-db.stanford.edu/LIC/LIC.html
0.110514	http://www-db.stanford.edu/LIC/mediator.html
0.032421	http://www-db.stanford.edu/people/
0.031012	http://www-db.stanford.edu/~gio/
0.018945	http://www-db.stanford.edu/~wangz/
0.018688	http://www-db.stanford.edu/~jan/watch/intro.html
0.016727	http://www-db.stanford.edu/tsimmis/
0.016446	http://www-db.stanford.edu/~danliu/
0.015506	http://www-db.stanford.edu/cs347/
0.015412	http://www-db.stanford.edu/LIC/
0.015224	http://www-db.stanford.edu/~widom/
0.014942	http://www-db.stanford.edu/~wilburt/
0.014096	http://www-db.stanford.edu/~chenli/
0.013814	http://www-db.stanford.edu/~ullman/
0.012856	http://www-db.stanford.edu/CHAIMS/
0.012687	http://www-db.stanford.edu/~echang/
0.012687	http://www-db.stanford.edu/~cyw/
0.012687	http://www-db.stanford.edu/~crespo/
0.012687	http://www-db.stanford.edu/~cho/
0.012405	http://www-db.stanford.edu/~sergey/

Note that most of these pages are actual home pages and entry pages to the lower level logical domains. This was due to the fact that only these home pages are relevant to the focused keywords *papers* and *publications* and their edges are assigned with a lower cost. Thus, the algorithm prefers to ”walk” through these pages over other 1400 pages. Some pages, such as `www-db.stanford.edu/LIC/` and `www-db.stanford.edu/LIC/mediator.html`, in the root logical domain are included due to its high connectivity.

Next, the algorithm recursively visited the nodes in this domain. Here we report on one of the largest subdomains, i.e., *http://www-db.stanford.edu/g̃io/*, (793 pages). When summarized with the proposed algorithm, the resulting graph contained the following pages:

Ranking	URL
0.015642	http://www-db.stanford.edu/pub/gio/CS545/image.html
0.011731	http://www-db.stanford.edu/pub/gio/1999/Interopdocfigs.html
0.011405	http://www-db.stanford.edu/pub/gio/biblio/master.html
0.011356	http://www-db.stanford.edu/pub/gio/CS99I/library.html
0.010754	http://www-db.stanford.edu/pub/gio/1994/vocabulary.html
0.009695	http://www-db.stanford.edu/pub/gio/CS99I/ubi.html
0.009483	http://www-db.stanford.edu/pub/gio/CS99I/entedu.html
0.009206	http://www-db.stanford.edu/pub/gio/CS99I/health.html
0.008717	http://www-db.stanford.edu/pub/gio/CS99I/wais.html
0.007642	http://www-db.stanford.edu/pub/gio/CS99I/security.html
0.006029	http://www-db.stanford.edu/pub/gio/CS99I/refs.html
0.005988	http://www-db.stanford.edu/pub/gio/gio-papers.html
0.005255	http://www-db.stanford.edu/pub/gio/inprogress.html
0.005010	http://www-db.stanford.edu/pub/gio/
0.003951	http://www-db.stanford.edu/pub/gio/paperlist.html
0.003924	http://www-db.stanford.edu/pub/gio/CS99I/description.html
0.003910	http://www-db.stanford.edu/pub/gio/CS99I/background/Cairncross97.html
0.003788	http://www-db.stanford.edu/pub/gio/CS545/
0.003299	http://www-db.stanford.edu/pub/gio/CS99I/copyright.html
0.002811	http://www-db.stanford.edu/pub/gio/CS545/indexing/

Note that these pages are either publication oriented pages, or they are linked to many pages with publication content: actual publication pages are omitted from the summarization to give place to pages which connect to many publication pages, allowing easier access to more information while browsing the Web. This is consistent with the goal of the topic distillation. Our second observation is that the pages with more recent course or publication information are ranked higher than those with older information. When we selected the pages relevant to ”publication” or ”papers”, we restrict the search on the pages modified after 1999. By considering freshness of information, we select up to date pages over those are not. Based on these experimental results, we believe the application of the random walk algorithm to Web site summarization is promising and results are useful.

5 Related Work

In this section, we summarize existing work related to the work presented in this paper, which integrates content search with structure analysis for query result space organization.

Researchers in the AI community have developed web navigation tour guides, such as WebWatcher[15]. WebWatcher utilizes user access patterns in a particular web site to recommend users proper navigation paths for a given topic. User access patterns can be incorporated into the random walk-based algorithm to improve the document association mining.

One approach to organizing web query results is “topic distillation” 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. An 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 reinforcing: 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, thus, it provides an effective mean for summarizing query results.

Bharat and Henzinger [16] improved the basic topic distillation algorithm [17] through additional heuristics. Fir example, the modified topic distillation algorithm considers only those pages that are in different domains for mutual authority/hub reinforcement. Preliminary experiments show improved results by introducing similarity measurement of linked documents. Another major variation of the basic topic distillation algorithm is proposed by Page and Brin[18]. Their algorithm further considers page fanout in propagating scores.

Topic distillation has been used by many search engines, including *Google*[19], NEC NetPlaza[20], IBM Clever[17], and TOPIC[21]. Many of the topic distillation algorithms have been also used to identify latent web communities[5, 22]. 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.

Dean and Henzinger[1] proposed two algorithms, *companion* and *co-citation* to identify related pages and compared their algorithms with the algorithm[2] used to implement the `What's Related?` functionality of Netscape. By extending the scope from documents to web sites, Bharat and Broder[23] conducted a study that compares 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 focuses on finding related documents or web sites. Our work focuses on finding pages inducing associations of given seed URLs.

Li et. al[24] present a technique for constructing multi-granularity and topic-focused site maps. Their technique has the following advantages: it is (1) capable of summarizing the contents of the web site and visualizing the topology of the web site, thus supports navigation; (2) flexible to present the overview of the contents and topology using multiple granularity; and (3) content-sensitive to support users with different interests. The criteria for selecting logical domain entry pages to form site maps are automatically adjusted by machine learning techniques. In this technique, the section of important pages are based on a set of user defined rules.

Note that *topic distillation* [3, 16, 18] could be a natural choice for summarization purposes. However, we observe that the behavior of the topic distillation algorithm may not be as good as our *document*

association based approach in the scope of summarization tasks:

- Topic distillation algorithm aims at selecting a small subset of the most “authoritative” pages and ”hub” pages from a much larger set of query result pages. An 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 reinforcing: good authoritative pages are linked by a large number of good hub pages and vice versa. Because the important pages are mutually reinforcing, the results tend to form a cluster. For example, there are many on-line HTML papers at `www-db.stanford.edu`. These papers consist of a number of pages linking to each other. By using the topic distillation algorithm, most of these pages are selected while many individual home pages are left out; this is not suitable for the purposes of summarization. On the other hand, our algorithm uses the concept of *seed nodes* which focus the summarization process, based on the web site structure as well as the user focus. Nodes selected after summarization are those nodes that explain why the seed nodes are related.

- The distance from the seed pages is not accounted for in topic distillation. Our algorithm, on the other hand, uses this distance value explicitly to assign edge weights that capture not only connectivity (as in the case of topic distillation), but also distance from the seeds. Note that, given a neighborhood, independent of the seed pages, topic distillation would return the same set of hubs and authorities.

- The original topic distillation algorithm does not account for the page contents once the initial pages are identified by an index search. Later works [16] introduce the document content into the topic distillation process. The main techniques introduced in this work is to use the query to document similarity to weigh nodes and prune those that seem to be irrelevant. Our algorithm, on the other hand works, by assigning edge probabilities based on the content of the pages; i.e., by guiding the random walk process based on the content of the pages to be visited. Therefore, our algorithm incorporates, connectivity, content, as well as distance information in a uniform manner.

Therefore a good choice of seed nodes (in our case, the entry-nodes of logical domains) leads into a good and meaningful summarization. Thus, a document association based approach is more suitable for the purpose of summarization.

6 Concluding Remarks

Recently, attentions have been paid to finding web page associations for the purpose of providing the user with additional pages which are relevant to the page the user is currently viewing. Examples of this

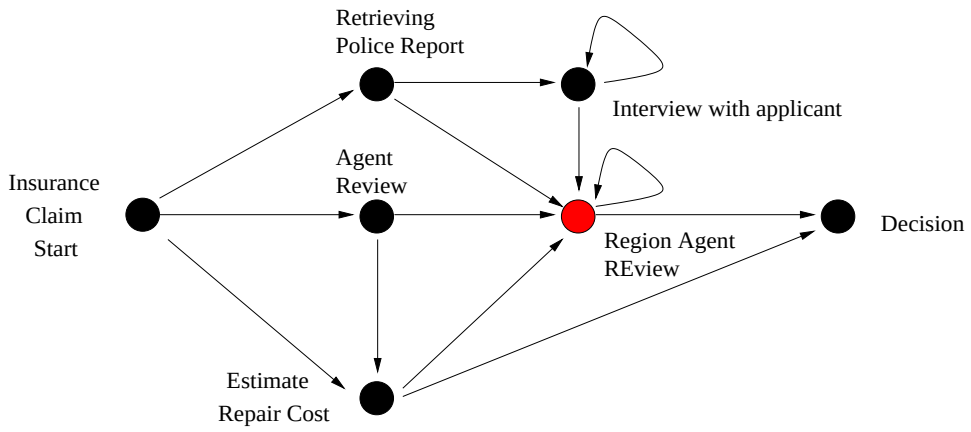


Figure 12: Random work algorithm application in workflow bottleneck identification

functionality include *what's related!* in Netscape and *companion pages* in AltaVista. On the other hand, we are interested "reasoning" web document associations. That is, "for a given set of pages, find out *why* they are associated".

In this paper, we present a framework for reasoning web document association. We assume the reasons of web page association are implicitly embedded in the links connecting these pages. Our reasoning scheme is based on a *random walk algorithm*. The algorithm has three major advantages: (1) the results by link analysis are derived based on equation solving rather than iteration used in other topic distillation methods; (2) the algorithm can take both link structure and contents into consideration; and (3) the algorithm allows users to specify a set of pages to focus for reasoning their associations. We validate the effectiveness of the algorithm on both synthetic and real web data. In the second part of the paper, we present how the algorithm can be combined with a developed logical domain identification technique for web site summarization as well as site map construction.

Future work will include extending this work for workflow management. In Figure 12, we show an example of insurance claim workflow. The workflow can be represented using our model and our algorithm can be useful to identify bottleneck of the workflow with specific focus. In this example, we identify the region agent review is the bottleneck of the claim process.

Acknowledgments

The authors would like to express their appreciations to the web site `www-db.stanford.edu` for its data used in their experiments. The reason why we have selected this web site is due to (1) our familiarity with the contents of the site, which helps in evaluating the results and (2) the fact that the pages in this site are not dynamically generated. The experimental results presented in this paper are for the purposes

of scientific research only.

The authors would like to express their appreciations to Quoc Vu, Okan Kolak, Necip Fazil Ayan, and Hajime Takano for their discussion, comments, and contribution to the work on the logical domain identification.

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. *ACM Computing Survey*, January 2000.

[5] 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.

[6] R. Lempel and S. Moran. The Stochastic Approach for Link-Structure Analysis (SALSA) and the TKC Effect. In 9^{th} *International World Wide Web conference*, Amsterdam, Holland, May 2000.

[7] R. Lempel and S. Moran. SALSA: The Stochastic Approach for Link-Structure Analysis. *ACM Transactions on Information Systems*, 19(9):131–160, April 2001.

[8] Frank K. Hwang, Dana S. Richards, and Pawel Winter, editors. *The Steiner Tree Problem (Annals of Discrete Mathematics, Vol 53)*. 1992.

[9] S.L. Hakimi. Steiner's problem in graphs and its implications. *Networks*, 1:113–131, 1971.

[10] Wired Digital Inc. *Information available at http://www.hotbot.com/*.

[11] Yahoo Inc. *Information available at http://www.geocities.com/*.

[12] America Online, Inc. *Information available at http://www.aol.com/*.

[13] NEC Corporation. *Information available at http://www.biglobe.ne.jp/*.

[14] Wen-Syan Li, Okan Kolak, Quoc Vu, and Hajime Takano. Defining Logical Domains in a Web Site. In *Proceedings of the 11th ACM Conference on Hypertext*, pages 123–132, San Antonio, TX, USA, May 2000.

[15] T. Joachims, D. Freitag, and T. Mitchell. Webwatcher: A tour guide for the world wide web. In *Proceedings of the 1997 Internaltional Joint Conference on Artificial Intelligence*, August 1997.

[16] 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.

[17] 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.

[18] 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.

[19] Google Search Egnine. *Information available at http://google.stanford.edu/*.

[20] NEC Corporation. NetPlaza Search Engine. *Information available at http://netplaza.biglobe.ne.jp/*.

[21] Davood Rafiei and Alberto Mendelzon. What is this Page Known for? Computing Web Page Reputations . In *Proceedings of the 9th World-Wide Web Conference*, Amsterdam, the Netherlands, May 2000.

[22] 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.

[23] 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.

[24] Wen-Syan Li and Necip Fazil Ayan and Okan Kolak and Quoc Vu and Hajime Takano and Hisashi Shimamura. Constructing Multi-granular and Topic-focused Web Site Maps. In *Proceedings of the 10th World-Wide Web Conference*, HongKong, China, May 2001.

Author Bibliography

Kasım Selçuk Candan is a tenure track assistant professor at the Department of Computer Science and Engineering at the Arizona State University. He joined the department in August 1997, after receiving his Ph.D. from the Computer Science Department at the University of Maryland at College Park. His dissertation research concentrated on multimedia document authoring, presentation, and retrieval in distributed collaborative environments. He received the 1997 ACM DC Chapter award of Samuel N. Alexander Fellowship for his Ph.D. work. His research interests include development of formal models, indexing schemes, and retrieval algorithms for multimedia and Web information and development of novel query optimization and processing algorithms. He has published various articles in respected journals and conferences in related areas. He received his B.S. degree, first ranked in the department, in computer science from Bilkent University in Turkey in 1993.

Wen-Syan Li is a Senior Research Staff Member at Computers & Communications Research Laboratories (CCRL), NEC USA Inc. He received his Ph.D. in Computer Science from Northwestern University in December 1995. He also holds an MBA degree. His main research interests include content delivery network, multimedia/hypermedia/document databases, WWW, E-Commerce, and information retrieval. He is leading CachePortal project at NEC USA Venture Development Center and Content Awareness Network project at NEC CCRL in San Jose. Wen-Syan is the recipient of the first NEC USA Achievement Award for his contributions in technology innovation.