Node Ranking In Labeled Directed Graphs

Krishna P. Chitrapura
IBM India Research Lab
Block-I, Indian Institute of Technology
Hauz Khas, New Delhi, India
kchitrap@in.ibm.com

Srinivas R. Kashyap [*]
Department of Computer Science
University of Maryland, College Park,
MD, USA
raaghav@cs.umd.edu

ABSTRACT

Our work is motivated by the problem of ranking hyper-linked documents for a given query. Given an arbitrary directed graph with edge and node labels, we present a new flow-based model and an efficient method to dynamically rank the nodes of this graph with respect to any of the original labels. Ranking documents for a given query in a hyper-linked document set and ranking of authors/articles for a given topic in a citation database are some typical applications of our method. We outline the structural conditions that the graph must satisfy for our ranking to be different from the traditional *PageRank*.

We have built a system using two indices that is capable of dynamically ranking documents for any given query. We validate our system and method using experiments on a few datasets: a crawl of the IBM Intranet (12 million pages), a crawl of the www (30 million pages) and the DBLP citation dataset. We compare our method to existing schemes for topic-biased ranking that require a classifier and the traditional *PageRank*. In these experiments, we demonstrate that our method is well suited for fine-grained ranking and that our method performs better than the existing schemes. We also demonstrate that our system can obtain an improved ranking with very little impact on query time.

Categories and Subject Descriptors

H.3 [**Information Systems**]: Information Storage And Retrieval

General Terms

Algorithms, Experimentation, Theory

Keywords

Search, Context-Sensitive Ranking, Search in Context, Web Graph, Intranet Search, Citation Graph, link structure, Flow-based Model, Random Surfer Model, PageRank

[*]Work done when the author was visiting IBM India Research Lab.

1. INTRODUCTION

The problem of ranking hyper-linked documents based on link information is very well studied [16, 10, 14, 18]. Any such scheme naturally finds an application in www search. The main ideas in the methods that have been proposed to solve this problem are based on the observation that links between documents often represent relevance [11] or confer authority [14, 3, 16, 11]. Further, it is assumed that a "reputed" document confers more relevance/authority to a document by linking to it than a less "reputed" document would if it were to link to the same document. These ideas have been explored in depth in the context of social networks [9, 4, 13, 12], citation analysis [17] and more recently in the context of web-scale information retrieval algorithms [11].

Ranking algorithms can be broadly classified into *query independent* ranking schemes and *query dependent* ranking schemes. Query independent ranking schemes like the *PageRank* assign a score to a document once and use this to order results for all subsequent queries, while query dependent ranking schemes assign a score that measures the quality and relevance of a page with respect to the given query. In this work, we are interested in *query dependent* ranking.

Traditionally, owing to their algorithmic simplicity, *query independent* ranking schemes have been made to work and scale successfully on the www. However, most existing *query dependent* ranking schemes involve the construction of a query-specific graph on which further analysis is then performed. The need for the construction of a query-specific graph during query-time tends to make these schemes infeasible for dynamic ranking on a web-scale.

We propose a new model and present an efficient method that affords dynamic (i.e. at query-time) *query dependent* ranking. We analytically prove the convergence properties of our method. Further, we validate our model and method by conducting experiments on a system that embodies our method and obtain improved performance over existing schemes, with very little impact on query time.

In subsection 1.1 we provide an additional layer of detail regarding our contributions and in subsection 1.2 we summarize previous work.

1.1 Our Contributions

We propose a new flow-based model for ranking documents biased by edge and node labels; and an efficient method for dynamically ranking for any given label. In our model, there is a set of flow values at each node associated with each label l. These flow values can be used to measure the importance of that node for the label l. We now consider one such label l. At each time step, every node accepts flow from its incoming edges and disperses its old value to its neighbors through its outgoing edges. If a node does not contain label l and has no incoming edges, the node never receives any flow. If a node has no outgoing edges, its old value is lost from the system (i.e., the node simply discards the old value and accepts the new value). At each time step, for each edge bearing the label l some value of flow (say f) is added to the *new* value of the node to which this labeled edge points. Similarly, at each time step, for each node bearing the label l some value of flow (say f) is added to the *new* value of this node. Typically, nodes are started off with zero values and these labels introduce values into the network.

This flow network will evolve over time. The value of the total flow in the network will increase with time and in some cases (conditions mentioned later) the flow values at all the nodes will saturate. In some other cases, when flow is conserved, the flow value in the system increases without bound. However in both cases, we can determine several useful quantities by observing the flow values for a label l at a node n_i.

As formalized in section 2.1, our method can also be viewed as a random surfer if we consider each node in G as a transient state.

For an arbitrary graph, depending on the structure of the graph, our method will either converge to a label-biased ranking that is different from *PageRank* or it will find the traditional *PageRank*. We discuss these conditions in more detail in section 2.2; but we simply note here that the web-graph, as presently known [6], has a structure that induces our algorithm to converge to a label-based ranking that is different from traditional *PageRank*. Not only is our ranking different from *PageRank*, but also through experiments we show in section 5.2.1 we show that our method provides a ranking that is orders of magnitude better than traditional *PageRank* (see table 1 and section 5.2.1). We experimentally show that our method is suited for fine-grained ranking by comparing our ranking with the ranking produced by topic-sensitive *PageRank* (see table 2 and section 5.2.2). Topic-sensitive *PageRank* is a *query-dependent* ranking scheme proposed in [10]. We also show through experiments (see section 5 and plots 9, 10 and 11) that these improved rankings can be obtained dynamically (i.e. low querying-time).

We have built a system (Section 4) using two indices that is capable of dynamically ranking documents for any given query. In Section 5, we validate our system and method on three datasets: a crawl of the IBM Intranet (12 million pages), a crawl of the www (30 million pages) and the DBLP citation dataset.

Although the main focus of our method is ranking hyper-linked documents in the www context, our method can be applied to other domains like citation analysis. To support this claim, we have performed experiments on the DBLP dataset (see section 5.2.3) and we provide anecdotal evidence that our method is suited for this domain as well.

1.2 Previous Work

There has been a tremendous amount of work related to the structure of information networks in the context of the www - *PageRank* [16], HITS [14] and the use of text to augment hyper-link information [15, 3, 8, 18, 10]. One cannot do justice to the amount of work in this area in anything less than a survey article and therefore we point the reader to Henzinger's survey article [11] for a review of existing techniques in the context of the www or see Friedkin's article [9] for insight into related methods used in the context of social networks and citation analysis.

Marchiori [15] proposed a structural approach of node-to-node hyper-information propagation. The scheme however does not consider cycles in the hyper-linked graph.

Many of the ranking schemes proposed are based on the random surfer model, where a web surfer's navigation is modeled as a Markov Process - i.e. The web surfer's decision on which page to go to next is based solely on the current page on which the surfer is located. In this model, each document in the web represents a state and the links going out the document represent possible transitions to other states. We consider some of the applications of this model below.

Page et al. [16] proposed a ranking measure called *PageRank* based on the random surfer model. In their model, the random surfer also occasionally jumps to a document chosen uniformly at random from the set of all documents (this is called a random jump). The *PageRank* (or the importance) of a document is the stationary probability of finding the surfer at that document. The algorithm is guaranteed to converge to a unique stationary probability iff G (the document graph) is strongly connected and the transition matrix of the random surfer is aperiodic. Dead-end documents (outdegree zero nodes in G) are a threat to the first requirement. Random jumps are introduced in *PageRank* to circumvent problems caused by dead-end documents as well as cycles. However, random jumps do not fully solve the problem caused by dead-end documents since the transitions out of dead-end nodes are still partially defined. There are, however, some *heuristics* that make structural modifications to G to try and overcome this problem.

The *PageRank* of a document is independent of label or topic choice. The idea of using a basis set of bias vectors to make *PageRank* query and user-sensitive was suggested in [5, 10]. Our method is closely related to [10] and this relationship is considered in detail later in this section.

The HITS algorithm proposed by Kleinberg [14] can be used to refine query results. However, it requires access to a search engine which will return a subset of the documents (deemed relevant to a query by the search engine). Chakrabarti et. al. [7] and Bharath and Henzinger [3] consider adjusting edge weights, using content analysis, for HITS to retrieve documents *related* to a query topic.

Rafiei and Mendelzon [18] present a search process that when given the URL of a document, returns a ranked list of topics on which that document has a reputation. They propose two models; a one-level model that extends *PageRank* and a two-level model that generalizes HITS. Their method works both in the presence and absence of a crawled dataset. However, their method (like HITS) requires access to search engine results. Our method requires access to a crawled dataset, but does not require access to a search engine.

Tomlin [19] introduced a new entropy maximization approach for measuring www traffic. The algorithm presented in the paper obtains a "traffic" and a local "temperature" which may be used for ranking pages. However, these measures are independent of the query terms or labels of interest.

Our method is closely related to the method of Haveliwala [10]. His method computes topic sensitive *PageRank* (or biased *PageRank*) by biasing the *PageRank* computation with respect to 16 representative topics taken from the Open Directory. It then classifies the query to these topics to compute the final weighted *PageRank*. The method uses a two-step ranking process. The first step is done offline and the second is done during query processing. During the first step, for each document the method precomputes a set of importance scores with respect to a set of chosen topics. In the second step, during query time, the method determines the broad topic of the query and based on this classification it combines the multiple precomputed scores to form a composite *PageRank* score.

The success of the first step of the scheme relies on the set of representative topics chosen initially and the quality of the ODP data. To guarantee convergence of the first step, the method has to make some structural modifications such as adding a complete set of edges to all the outdegree zero nodes in the document graph G. The second step involves the use of a classifier and this must be very well tuned to the queries the system handles in order for the scheme to perform well. As we show later in Section 5 this method performs well for broad-topic rankings. However, it is not well suited for fine-grained topic rankings.

Our method is closely related the measure of sociometric status proposed by Hubbell [12].

We will compare our results primarily with static *PageRank* [16] and the topic-sensitive *PageRank* method proposed in [10].

2. THE MODEL

In this section we formalize our flow-based model, relate our model to a random surfer model, analyze the convergence properties of our algorithm and provide structural conditions that G must satisfy for the flow values computed by our model to be different from traditional PageRank.

We are given a labeled directed graph $G(V, E, L_V, L_E)$, where V is the set of nodes, E is the set of edges, L_V is a label function that maps nodes to labels ($L_V : V \to \{labels\}$) and L_E is a label function that maps edges to labels ($L_E : E \to \{labels\}$). In a typical www setting, V is a set of hyper-text documents, E is the set of hyper-links connecting the documents in V, the edge-label is the anchor-text corresponding to a hyper-link and the node-label is the title of the document. Our flow model (briefly outlined in 1.1) can be realized by an iterative scheme. Define a vector $\mathbf{v}_l$ as:

$$v_l[i] = \sum_{j:(j,i)\in E} f_{(j,i,l)} + f_{(i,l)} \qquad (1)$$

where $f_{(j,i,l)}$ and $f_{(i,l)}$ are scalars corresponding to the amount of flow introduced at the corresponding labeled edge and labeled node respectively at each iteration. In equation 1 above, $f_{(j,i,l)} > 0$ if $L_E(j,i) = l$ and $(j,i) \in E$. $f_{(j,i,l)} = 0$ otherwise. Similarly, $f_{(i,l)} > 0$ if $L_V(i) = l$ and $i \in V$. $f_{(i,l)} = 0$ otherwise.

The vector $\mathbf{y}_{t+1}$ of flow values at the nodes at time $t+1$ is computed by iterating the following equation:

$$\begin{aligned} y[i]_{t+1} &= v_l[i] + \textstyle\sum_{j,(j,i)\in E} \frac{1}{outdegree(j)} y[j]_t \\ \mathbf{y}_0 &= \mathbf{0}_{n\times 1} \end{aligned}$$

Definition 1. *Define:*

$$W_G = \begin{bmatrix} w_{11} & \dots & w_{1n} \\ \vdots & \vdots & \vdots \\ w_{n1} & \dots & w_{nn} \end{bmatrix}$$

as the $n\times n$ matrix for G. Set $w_{ij} = \alpha/n + (1-\alpha)(1/outdegree(j))$ if $(j,i) \in E$ and α/n otherwise. Here $\alpha \in [0,1]$ is a scalar value.

The iteration above can be succinctly expressed as follows:

$$\begin{aligned} \mathbf{y}_{t+1} &= \mathbf{v}_l + \mathbf{W}_G \mathbf{y}_t & (2) \\ &= \mathbf{v}_l + \mathbf{W}_G \mathbf{v}_l + \ldots + \mathbf{W}_G^{t+1} \mathbf{v}_l & (3) \end{aligned}$$

where all nodes are started off with zero values, $\mathbf{W}_G$ is an appropriately defined $n \times n$ matrix (set $\alpha = 0$ in Definition 1) and vector $\mathbf{y}_{t+1}$ corresponds to the flow values at the nodes after $t+1$ iterations.

Note: In general, we can define $\mathbf{W}_G$ to be any matrix such that the column-sums corresponding to out-degree zero nodes in G are less than 1. The column-sums corresponding to all other nodes are equal to 1. That is, for all columns j:

- $\sum_i w_{ij} = 1$ if *outdegree*$(j) \geq 1$.
- $0 \leq \sum_i w_{ij} < 1$ if *outdegree*$(j) = 0$.

2.1 Relationship to Random Walker Models

There is an interesting relationship between our model and Markov Processes. Consider a Markov Process whose transition matrix is given by:

Definition 2.

$$\hat{\mathbf{W}} = \begin{bmatrix} \mathbf{W}_G & \mathbf{0} \\ \mathbf{R} & 1 \end{bmatrix}_{(n+1)\times(n+1)}$$

The $\mathbf{W}_G$ submatrix of $\hat{\mathbf{W}}$ is the same as the $n \times n$ $\mathbf{W}_G$ matrix in Definition 1. $\mathbf{R}$ is a $1 \times n$ matrix suitably chosen so that $\hat{\mathbf{W}}$ is column-stochastic. The last column of $\hat{\mathbf{W}}$ has zeros in rows 1 through n and a one in row $n+1$.

Consider each node in G to be a state in a Markov Process. We have essentially added an absorbing state to the system (i.e. a state which once entered cannot be exited). Note that if $\mathbf{W}_G$ were column stochastic to begin with, the system will never enter the newly added absorbing state. However, if $\mathbf{W}_G$ had some states such that transitions out of these states were partially defined, then all such partially defined states can reach the newly added absorbing state. Moreover, if all the nodes in $\mathbf{W}_G$ could reach one or more of these partially defined states, then the system will eventually end up in the newly added absorbing state. In case all the nodes in $\mathbf{W}_G$ can reach the newly added absorbing state, the following facts are well known from Markov theory:

- The nodes of $\mathbf{W}_G$ represent *transient* states (Any state that is not absorbing is called transient).
- The (i,j) entry of the quantity $(\mathbf{I} - \mathbf{W}_G)^{-1}$ (also called the *fundamental matrix*) is the expected number of time periods spent in node i before reaching an absorbing state given that the process was started off in node j.

- $\lim_{p\to\infty} \hat{\mathbf{W}}^p = \begin{bmatrix} \mathbf{0}_{n\times(n+1)} \\ \mathbf{1}_{1\times(n+1)} \end{bmatrix}_{(n+1)\times(n+1)}$

  (where, as usual, $\hat{\mathbf{W}}^p$ is the p^{th} power of the $\hat{\mathbf{W}}$ matrix in Definition 2).

2.2 Structure and Convergence

The following theorem establishes a sufficient condition for the flow values at the nodes in the network to saturate.

Theorem 1. *If every node in G can reach an out-degree zero node in G using transitions defined in $\mathbf{W}_G$, then the flow values in network G will saturate.*

Proof. Define matrix $\hat{\mathbf{W}}$ as in Definition 2. The matrices $\mathbf{R}$ and $\mathbf{W}_G$ have the same definition as in Definition 2. Now, if all the nodes in G can reach an out-degree zero node in G using transitions in $\mathbf{W}_G$, then all states (nodes) in $\mathbf{W}_G$ are transient and can reach the newly added absorbing state in $\hat{\mathbf{W}}$. If all the transient states (nodes) in $\mathbf{W}_G$ can reach the newly added absorbing state, we know from Markov theory that:

$$\lim_{p\to\infty} \hat{\mathbf{W}}^p = \begin{bmatrix} \mathbf{0}_{n\times(n+1)} \\ \mathbf{1}_{1\times(n+1)} \end{bmatrix}_{(n+1)\times(n+1)}$$

It can also easily be verified that:

$$\hat{\mathbf{W}}^p = \begin{bmatrix} \mathbf{W}_G^p & \mathbf{0} \\ \mathbf{Q} & 1 \end{bmatrix}$$

where $\mathbf{Q}$ is some $1 \times n$ matrix. So $\lim_{p\to\infty} \mathbf{W}_G^p$ is the zero matrix. Therefore, from Eqn. (3) we know that $\lim_{p\to\infty} (\mathbf{y}_{p+1} - \mathbf{y}_p) = 0$. That is the amount of flow at any node will remain constant from one iteration to the next. □

Setting $\alpha > 0$ is like adding random jumps to G. Random jumps essentially enable the algorithm to provide label-biased rankings on a larger class of graphs.

We will now consider the case when the flow values in G do not saturate. We know from Markov theory that if the transition matrix $\mathbf{W}_G$ for a Markov process is regular (i.e., some power of $\mathbf{W}_G$ has only positive entries), then there exists a unique steady state distribution for the process. For regular $\mathbf{W}_G$, we also know that the Markov process reaches this unique steady state distribution irrespective of the initial probability distribution. *PageRank* essentially relies on these properties of regular Markov systems to obtain the steady state distribution for the nodes in G. When the transition matrix $\mathbf{W}_G$ is regular, this steady state distribution also corresponds to the principal eigenvector of $\mathbf{W}_G$. The definition below codifies these facts:

Definition 3. *G is* PageRank*able, i.e., PageRank is well defined for G, if for the corresponding $\mathbf{W}_G$:*

$$\lim_{p\to\infty} \mathbf{W}_G^p = \begin{bmatrix} c_1 & \dots & c_1 \\ \vdots & \vdots & \vdots \\ c_n & \dots & c_n \end{bmatrix}_{n\times n}$$

The vector $\mathbf{c} = \begin{bmatrix} c_1 \\ \vdots \\ c_n \end{bmatrix}$ *is a principal eigenvector of* $\mathbf{W}_G$ *with an associated eigenvalue of* 1*, where* $\| \mathbf{c} \|_1 = 1$.

The lemma below shows that the total flow value in the system will increase without bound if G is *PageRank*able.

Lemma 1. *If G is* PageRank*able, then the flow values in network G will* not *saturate.*

Proof. From Definition 3 we know that if G is *PageRank*able, then $\lim_{p\to\infty} \mathbf{W}_G^p \neq \mathbf{0}_{n\times n}$. Therefore the vector of flow values will continue to increase from one iteration to the next and the flow values in network G will not saturate. □

The change in flow at node i from iteration t to iteration $t+1$ is $y[i]_{t+1} - y[i]_t$, where $\mathbf{y}_t$ and $\mathbf{y}_{t+1}$ are the vectors of flow values at the nodes after t and $t+1$ iterations respectively. The lemma below shows that when G is *PageRank*able, the change in flow at any node can be used to find *PageRank*.

Lemma 2. *For* PageRank*able G, the quantity $\frac{y[i]_{t+1} - y[i]_t}{||\mathbf{v}_l||_1}$ will equal* PageRank(i)*, for sufficiently large t.*

Proof. The change in flow values at the nodes is given by $\mathbf{y}_{t+1} - \mathbf{y}_t = \mathbf{W}_G^{t+1}\mathbf{v}_l$. For *PageRank*able G, from Definition 3 we know that $\lim_{t\to\infty} \mathbf{W}_G^p = \begin{bmatrix} c_1 & \dots & c_1 \\ \vdots & \vdots & \vdots \\ c_n & \dots & c_n \end{bmatrix}_{n\times n}$. So for sufficiently large t, the quantity $\lim_{t\to\infty} \mathbf{W}_G^{t+1}\mathbf{v}_l = \lim_{t\to\infty} \mathbf{W}_G^{t}\mathbf{v}_l = ||\mathbf{v}_l||_1\mathbf{c}$. Where $\mathbf{c} = \begin{bmatrix} c_1 \\ \vdots \\ c_n \end{bmatrix}$ is the principal eigenvector of $\mathbf{W}_G$. Therefore, $y[i]_{t+1} - y[i]_t = \left(\mathbf{W}_G^{t+1}\mathbf{v}_l\right)[i] = ||\mathbf{v}_l||_1 \cdot c_i$. Where c_i is the *PageRank* of i i.e. the i^{th} entry of the principal eigenvector of $\mathbf{W}_G$. □

2.3 Implications and Remarks

- When G is *PageRank*able, our model can only find *PageRank*. However, experiments have shown that the Web graph has a bow-tie structure (see [6]). A graph with such a structure is **not** *PageRank*able due to the presence of a large number of out-degree zero nodes. The web graph as currently known [6] satisfies the sufficient condition for convergence outlined in Theorem 1. As a result, our method can be applied to the Web graph *per se*.

- Our method can be applied any arbitrary graph (provided we add random jumps) and our method is guaranteed to converge - it will either give us a label-biased ranking of the nodes or it will simply compute the label-independent ranking of the nodes (i.e. *PageRank*). Although our method is shown to converge in the limit of the number of iterations, in practice we have found that our algorithm converges in a few steps.

- When G satisfies the sufficient condition in Theorem 1, different choices of l will lead to different results. Our method can then be used for a variety of purposes; Searching, Classification of sites and focused advertising to name a few. However, in this paper we focus mainly only on ranking documents for a given label. Through experiments on the DBLP citation dataset, we demonstrate that our method can be applied to other domains as well.

3. THE METHOD

3.1 An iterative scheme for a fixed label

To find the flow values at all the nodes for a fixed label, we can use the following iterative scheme: Define a vector $\mathbf{v}_l$ as defined in equation 1.

The vector $\mathbf{y}$ of node ranks biased by edge-labels is then computed iteratively by the following equation:

$$\mathbf{y}_{t+1} = \mathbf{v}_l + \beta \mathbf{W}_G \mathbf{y}_t \qquad (4)$$

where $\beta \in [0, 1]$ and $\mathbf{W}_G$ is a suitably defined $n \times n$ matrix as in Definition 1. The β premultiplier is not required to ensure convergence if $\mathbf{W}_G$ satisfies the convergence criterion of Theorem 1. However, choosing a $\beta > 0$ speeds up convergence.

The algorithm seeks to find the steady state vector $\mathbf{y}_s$ for the iteration above.

$$\begin{aligned}\mathbf{y}_s &= \mathbf{v}_l + \beta \mathbf{W}_G \mathbf{y}_s \\ &= (I - \beta \mathbf{W}_G)^{-1} \mathbf{v}_l \\ &= \left(I + \beta \mathbf{W}_G + (\beta \mathbf{W}_G)^2 + (\beta \mathbf{W}_G)^3 + \ldots\right) \mathbf{v}_l \quad (5)\end{aligned}$$

In practice, the iterative algorithm declares a solution once $||\mathbf{y}_{t+1} - \mathbf{y}_t||_2 \leq \epsilon$, or when the ranks of the top few nodes remain unchanged from one iteration to the next.

The condition for the expected values at each node to stabilize is that the quantity $(I - \beta \mathbf{W}_G)^{-1}$ must exist. The quantity $(I - \beta \mathbf{W}_G)^{-1}$ is guaranteed to exist and our iterative algorithm is guaranteed to converge if $||\beta \mathbf{W}_G|| < 1$, where $||\beta \mathbf{W}_G|| = \beta \max_j \sum_i |w_{ij}|$. Since $\beta < 1$ and $\sum_i |w_{ij}| \leq 1$ for all i, $||\beta \mathbf{W}_G|| < 1$, our algorithm will converge. The β term acts like a damping factor and its value can be set as close to 1 as required so that $||(1 - \beta) \mathbf{W}_G|| \leq \epsilon$ (for instance choose $\beta \geq 1 - \frac{\epsilon}{n}$ for the Frobenius norm).

We **do not** use the iterative scheme for a fixed label presented here since we want to perform dynamic ranking. However, we do use the structure of the iteration in this scheme to compute a matrix that can be used to perform dynamic ranking. The scheme is presented in the next subsection.

3.2 Dynamic Ranking

The previous section provided a method for computing the steady state flows at all nodes for a fixed label. In practice, we need to compute the flows across all nodes over a large number of labels and it would be infeasible to follow the approach in the previous subsection. Therefore, it is preferable to compute a matrix which we call $\mathbf{B}$, that stores the reachability information through all possible paths between any pair of nodes in the graph.

The (i, j) entry in the reachability matrix encodes the total flow that node i observes given that a unit of flow is inserted at node j. This entry is a measure of the effect that the node j has on node i. This effect can be through paths of various lengths. If we fix the maximum length of influence paths (call it t_{max}) that we wish to consider, a good approximation to the reachability matrix can be found efficiently, provided t_{max} is small. Since n is *very* large and t_{max} is a small fixed number, we can ignore the effect of random jumps (i.e. transitions from node j to node i when $(j, i) \notin G$). Once we precompute such an approximate reachability matrix, given any label l we can quickly find a suitable vector $\mathbf{v}_l$ (see Sec 3.1) and compute node ranks for the given label. The main challenge in computing the $\mathbf{B}$ matrix is one of scale.

Let $\mathbf{B}$ to be a reachability measure where B_{ij} represents the total influence that node j exerts on node i. We can then define:

$$\mathbf{B} = \mathbf{I} + \beta \mathbf{W}_G + (\beta \mathbf{W}_G)^2 + (\beta \mathbf{W}_G)^3 + \ldots + (\beta \mathbf{W}_G)^{t_{max}} \quad (6)$$

Note that $\mathbf{I}$ represents influences through paths of length zero, $\mathbf{W}_G$ represents influences through paths of length one, and so on.

In practice, $\mathbf{B}$ is calculated using an iterative algorithm. The number of iterations of this algorithm corresponds to the maximum length of influence paths we wish to consider. Iteratively, we can write:

$$\begin{aligned}\mathbf{B}(0) &= \beta \mathbf{W}_G \\ B(t+1)_{ij} &= \beta \left(W_{ij} + \textstyle\sum_{(k,j) \in E, (i,k) \in \mathbf{B}} B(t)_{ik} W_{kj}\right)\end{aligned}$$

After the final iteration we add $\mathbf{I}$ to $\mathbf{B}$. To compute $\mathbf{B}(t+1)$, we need to store $\mathbf{B}(t)$ and $\mathbf{W}_G$.

The following lemma establishes the equivalence of the matrix $\mathbf{B}$ in (6) and the matrix $\mathbf{B}$ computed by the iterative algorithm above.

Lemma 3. *At the end of iteration t, any B_{ij} entry will correspond to the total influence of node j on node i through all paths of length 1 through t from node j to node i.*

This scheme can be realized by storing $\mathbf{B}$ and $\mathbf{W}_G$ as sparse indices, where for each node j we can query and obtain a list of nodes reachable/pointed to by that node. The underlying mechanism responsible for storing such a forward index might also prune some out-edges from time to time to meet storage constraints as the $\mathbf{B}$ matrix can be fairly dense.

Since our goal is to search for the top k nodes that best satisfy a query, we may not need the complete $\mathbf{B}$ matrix. We can store the top m reachable nodes for every node i at every iteration. Further, we show in the next few sections, that such an approximation helps improve the performance without a significant impact on the accuracy of the retrieved result.

3.3 Multiple term queries

To find the top k nodes for a single label l, we construct the biasing vector $\mathbf{v}_l$ for the label l, and then multiply the sparse matrix $\mathbf{B}$ with the vector $\mathbf{v}_l$. We pick the top k highest entries from the resulting vector and return the nodes corresponding to these entries as the k best search results for the label l. We handle multiple queries by constructing an appropriate biasing vector $\mathbf{v}_q$ using "fuzzy AND" and "fuzzy OR". The details are provided in the Section 4.

3.4 Query independent ranking

One can use our method to compute a query independent ranking, similar to *PageRank*, which we call non-biased ranking (NBR). Non-biased-Rank is obtained by uniform bias with each entry i of $\mathbf{v}_l$ set to some constant value, say $1/n$. This can be best visualized as a constant unit of flow being introduced at every node, and we observing the steady state flow at every node to be the rank of that node.

3.5 Comparison with existing methods

As outlined in Section 2, our method is closely related to other methods that are based on Markov chain model. Two such methods we discuss here are those proposed in [16, 10].

The *PageRank* method [16] solves an equation that is very similar to that of NBR incarnation of our method (see (4)), when $\mathbf{v}_l$ is set to uniform bias, i.e. labels are found uniformly at random on the graph, $[\frac{1}{n}]_{1\times n}$. Al-though, the methods used to solve the equations are different, the solutions will be similar. We empirically compare this ranking found by our method using non-biased ranking and compare it with traditional *PageRank* in section 5.

The topic-sensitive *PageRank* method [10] uses a different biasing vector for each topic. The biasing vector $\mathbf{p}_j$ for topic T_j is defined by

$$p_{ji} = \begin{cases} \frac{1}{|T_j|} & \text{when } i \in T_j \\ 0 & \text{otherwise} \end{cases} . \tag{7}$$

The bias vector $\mathbf{p}_j$ in very similar to $\mathbf{v}_l$ in our case, except for the fact that in our case $|\mathbf{v}_l| \neq 1$ and $\mathbf{v}_l$ is biased towards the documents that have more of label l in the anchor text of their inlinks. Further, we do not need a classified set of documents to set up our bias vector and can easily rank fine grained topics. Our method also eliminates the need to classify/disambiguate the queries before using the biased ranks. The most important difference is that our method provides dynamic ranking given a query belonging to any topic.

4. THE SYSTEM

Our system is implemented using two indices. The first is a reverse index which maps each label to a set of pages i, if the anchor text on the in-links of these pages contain that label. The mapping also stores the of value $v_l[i]$ and the values of the unmapped entries are assumed to be zero. The second index stores the sparse matrix representation of the $\mathbf{B}$ matrix which can look up all the pages j that are influenced by a particular page i, along with the value of B_{ij}. Non-existent j entries in the index mean that the corresponding nodes are not reachable by page i and such pages are assumed to have a reachable score of zero.

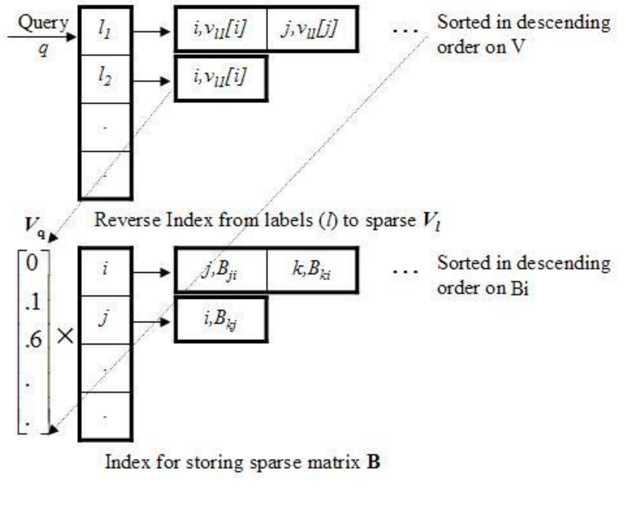
Query
q
l_1
l_2
$i,v_{l1}[i]$
$j,v_{l1}[j]$
$i,v_{l1}[i]$
Sorted in descending order on V
Reverse Index from labels (l) to sparse V_l
V_q
0
.1
.6
×
i
j
j,B_{ji}
k,B_{ki}
i,B_{kj}
Sorted in descending order on Bi
Index for storing sparse matrix **B**

Figure 1: Indices in the Search Engine

We stem the labels using the standard Porter's stemmer. We remove stop-words, but do not used any synonyms or thesauri. The indices are implemented as flat file indices on a standard Linux box (Red hat 7.1) with the Reiser file system. The machine is an IBM IntelliStation ZPro with dual Intel Pentium III CPU running at 1 GHz, Dual SCSI hard disks and a GB of RAM.

Given a Boolean query q which has labels and Boolean operands, as a first step, we look up the label index for the vector $\mathbf{v}_l$ for each of the labels in q and further form a vector $\mathbf{v}_q$ based on the following rules:

- if $q = l_1$ AND l_2 , then $\mathbf{v}_q$=$\mathbf{v}_{l_1} \perp \mathbf{v}_{l_2}$, which we have implemented as 'fuzzy AND', $v_q[i]$=$min(v_{l_1}[i], v_{l_2}[i])$.

- if $q = l_1$ OR l_2 , then $\mathbf{v}_q$=$\mathbf{v}_{l_1} \top \mathbf{v}_{l_2}$, which is 'fuzzy OR', $v_q[i]$=$max(v_{l_1}[i], v_{l_2}[i])$.

The second step is a sparse matrix multiplication of the vector $\mathbf{v}_q$ and the matrix $\mathbf{B}$. We use a straight-forward implementation in which each entry $v_q[i]$ is multiplied with the i^{th} row of the $\mathbf{B}$ matrix and the results are added in memory to form the vector $\mathbf{y}$ that contains the ranking of the documents for the query q. For the sake of performance, we keep the entries in $\mathbf{B}$ sorted on their magnitude, so that the top k results can be computed quickly. In theory, a sparse matrix-vector multiplication is $O(n^2)$. However, in practice, given the fact that we are interested in computing only the top k ranks, we can limit the number of entries in $\mathbf{B}$ to the top m to compute the ranks online. In section 5, we discuss this heuristic in detail and show that it is practical and scalable.

5. EXPERIMENTAL SETUP

5.1 Datasets

We have conducted experiments on two web based corpora and a citation database to demonstrate the utility as well as the scalability of our method.

5.1.1 Intranet Data

The first corpus contains a crawl of the IBM Intranet of about 12 million pages. We removed the references to pages that were not in the crawl but were linked by pages in the crawl. We used the anchortexts as the edge-labels. We did not have any node labels. We had about 150,000 unique labels in the system that occurred at least thrice after stemming. We also had access to 240 common queries and a primary result for each of the queries as judged by "experts" [19]. Some of the queries were repetitive with the only difference being the case (upper or lower), number (singular or plural), etc. We also cleaned up the answer URLs to handle duplicates, moved URLs, and auto-redirections. We also removed 39 query-answer pairs as the answer URLs were outdated and not present in the current crawl. After removing all such duplicate queries and non-existent answer URLs, we had 165 unique query-answer pairs. These queries were a mixture of *navigational* and *informational* type [1]. Since we do not have a full text index and are indexing the anchor-text, we classified the queries based on number of documents whose in-link's anchor-text contains the labels in the queries. The majority of the queries were navigational with 132 of them occurring in more than 10,000 anchors.

5.1.2 Internet Data

The second dataset is a 30 million page crawl of the Internet, starting from many popular media sites and music-relates sites. We indexed 167,000 unique stemmed labels from the anchor-text that occurred at least 10 times.

One of our goal is to compare our results with that of topic sensitive *PageRank* [10]. We resorted to the dmoz[1] to obtain a source of human-rated pages on topics. Considering, only the "Top/Arts" topic of dmoz, since it is one of the most populous and related to the crawl, we crafted an experiment to compare these two methods. We first spotted the pages in our crawl that belonged to the topic in question and then retained half of the spotted pages (ground truth) for "training". We use the training data to set the on-topic bias in case of the [10] and to build the vector $\mathbf{v}_q$ in our method. We used the remaining half as the test set where we observe their average rank as ranked by the method in [10] and our method.

To establish the ground truth (or golden data) for the URLs in our crawl, we had to map the URLs to the closest site/subsite in the dmoz directory. This was accomplished by sorting both our list of URLs and the URLs in the < link r:resource> tags of the dmoz RDF distribution and then performing a pair wise merging, by mapping URLs in our list to the closest URL in dmoz. The closeness is a simple lexicographical closeness of URLs based on whether they share a common domain name and a common directory structure. For example, *http://www.yahoo.com/index.html* is close to *http://www.yahoo.com* than to *http://mail.yahoo.com*. This exercise was necessary as the dmoz is directory of the Internet sites (not pages) and also could contain many sub-sites of a same physical site site (*http://www.geocities.com*) under different categories. We have used 2001 RDF distribution of dmoz and the crawl we have used was completed in early 2002. The number of pages spotted at each level are shown in the Table 2, where level 1 consists of all pages that fall under the topic "Top/Arts" and all its subtopics, level 2 consists of all pages that fall under the subtopics of topic "Top/Arts" (but not directly the topic itself), and so on.

5.1.3 DBLP Data

Digital Bibliography and Library Project (DBLP) dataset has details of publications in the CS domain in the recent years. We have used dblp XML distribution dated 9th of December 2003 [2], which has 457,261 articles, 171,133 citations and more than 300,000 unique authors. We constructed a graph with articles and authors as the nodes and citations as the edges. We also added directed edges from the article nodes to their author nodes (Figure 2). The articles were labeled with the text in the <title> tag and the edges were labeled with text in the label attribute of the <cite> tags. We also employed the standard Porter's stemmer to reduce the words in the labels to their base form. The dictionary we built on all these labels had close to 20,000 unique words, which occurred at least 10 times.

We obtained a few human edited list of researchers in computer science from
http://dmoz.org/Computers/Computer_Science/Theoretical/People/
http://dmoz.org/Computers/Algorithms/People/
http://theory.lcs.mit.edu/̃rajiyer/theory_folks.html .

[1]http://dmoz.org

[2]http://dblp.uni-trier.de/xml/

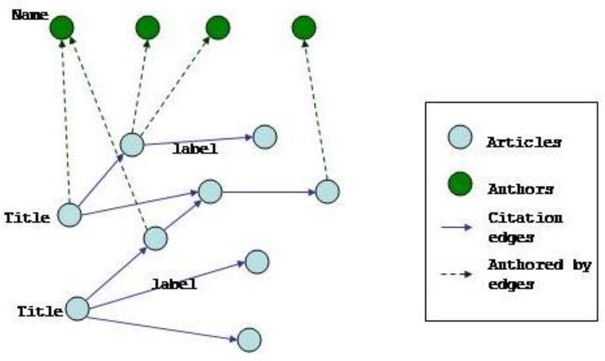

Figure 2: The structure of the DBLP graph

These contains a list of people working in theoretical computer science along with a line of description of their area. Our goal was to construct queries from the description line and observe the corresponding rank of the person as returned by our system. As these are human edited lists, we can assume that these are popular researchers in this particular area and should be ranked high. After merging these lists we had 92 unique people but some had more than one description of their area. So, in total we had 278 query(description) people pairs. We cleaned up the description not to include names of the university, department etc.

5.2 Results

Armed with the three datasets, their respective sets of queries and golden answers, we devised some experiments to compare our method with existing methods for ranking. We also conducted some experiments to measure the sensitivity of our system in terms of querying time and quality of result to the different types of query loads. Our experiment goal was to observe at which rank (1 through n) our system and the traditional ranking methods placed the golden answer for a particular query. We consider the average rank of the golden answer for all the queries as a measure of performance in all our experiments. If the query contains no terms in our label index, then we assume the ranking to be same as that given by the uniform bias, i.e., the NBR. If the golden answer does not appear in the ranked list, we assign it the max rank in the system (n).

Through out these experiments, we have used $\beta = 1$, no random jumps and the value of $f_{(j,i,l)}$ (defined in Sec. 2) to be $1/outdegree(j)$ corresponding to the probability of a random surfer going from node j to node i. Where ever we have not mentioned explicitly, the max number of entries in the $\mathbf{B}$ matrix is 100 and number of terms used to compute $\mathbf{B}$ matrix is 10.

5.2.1 Traditional PageRank

We used a standard iterative implementation of *PageRank* with random jumps [19] with the probability of a random jump being set to 0.1. In the results shown in this section, non-biased-Rank was obtained by computing static rank with $\mathbf{v}_l$ as uniform bias with each entry i set to $1/n$. The Edge-label biased ranking is obtained by our algorithm with $\mathbf{v}_l$ set to $\mathbf{v}_q$ for each of the queries given by the experts.

The results summarized in Table 1 show that edge-label

Table 1: Average rank of intranet test URLs on 165 queries

PageRank	Non-biased Rank	Edge-label biased rank
0.73×10^6	0.31×10^6	510

biased ranking has a *v*ery low average rank for the answers to the experts' queries in the intranet data. In fact, 62 of the 165 queries returned the most preferable answer within the top 10 ranks. 107 of the queries had the most preferable answer within the top 20 ranks. All the answers were present in the ranked lists, i.e., we never had to assume the max rank for any of the answers. However, eight of the queries had no matching labels, and we obtained the ranks for the corresponding answers by their position in the non-biased rank. The problem of non-matching labels is not very acute, and can be circumvented by using a full-text index and a classifier or a dictionary that finds the closest labels to the query. If we remove these 8 queries, the average rank further drops down to **28.43**. Similar observations, where label biased ranking outperforms *PageRank* can be made with respect to the Internet data (as summarized in table 2).

5.2.2 Topic sensitive PageRanking

The next task was to perform ranking using topic sensitive *PageRank* and our method. Topic sensitive *PageRank* needs a classified set of pages on the topics in question. As we do not have any classification on the intranet data, we compared our method to this method on the Internet data. In the case of topic sensitive ranking, we created a bias vector $\mathbf{v}$ as given in (7) where topic set is the training set T.

In the case of ranking biased by edge-label we inspect all the inlinks of the documents in T and collate all their anchor text to formulate $\mathbf{v}_q$; where q is OR of all labels $L(j,i)$ such that page $i \in T$

We computed the ranks of the test URLs for each subtopic by using both these methods and collated the average rank level-wise. For example, we computed the ranks of the test URLs under "Top/Arts/Music" by biasing the ranking with the training URLs under the same topic, but collated the average rank over all the topics in level 2, i.e. "Top/Arts/Movies", "Top/Arts/Television", etc. The number of spotted pages cited in Table 2 is the total number of pages that fall under the topic at the cited level. The total number of pages under level 1 and level 2 are the same, since level 1 has only one topic and does not have any URL directly assigned to it. However, there is a difference in average rank because the ranking is computed topic-wise and averaged level-wise.

The results in table 2 shows that pages lower in the dmoz tree have lower *PageRank* and further that biasing the rank based on the topic helps. Rank biased by edge-label does especially well for fine-grained ranking, which is exactly the primary requirement of a search engine. Topic sensitive *PageRank* performs well at broad topics (all document under "Top/Arts") but performs significantly worse than edge-label biased ranking at deeper levels of dmoz. We believe that the somewhat higher ranking by edge-label biased ranking at the broad topic level is due to the presence of a large number of anchors in the query.

Table 2: Average rank of WWW test URLs($\times 10^6$)

level	No. of page spots	PageRank	topic-sensitive PageRank	edge-label biased rank
1	132,428	4.128	0.11	0.32
2	132,428	4.128	0.18	0.208
3	102,568	3.94	0.18	0.141
4	87,202	3.91	0.24	0.126
5	73,812	3.78	0.41	0.112
6	33,067	3.55	0.71	0.08

5.2.3 DBLP: Anecdotes

There has been work [2] which provided keyword based search on the dblp data, with ranking schemes to rank articles and authors for a given query. Other commonly applied techniques is to rank authors by the number of publication in a particular area and/or the number of citations to their work (*http://citeseer.org/*). We start with some anecdotal results to argue against ranking the authors/articles by solely the number of publications/number of citations. We argue that the quality of the articles that cited this article are also important, analogous to the *PageRank* on the Web. As ranking has to be fined grained (very few authors write in articles in varied number of areas and very few articles are referred from outside its area) this dataset is suited for labeled biased ranking.

For the query "database", Michael Stonebraker is ranked 9th when considering just the number of his publications in the database area and number of citations to his publications, but moves up 2nd position after applying label biased ranking. Similarly, Jeffery Ullman moves to the 1st position from 6th. We observed similar movement of the rank in the case of queries such as "approximation algorithms" where David P. Williamson moves to the top of the list, "geometric algorithms" where Ketan Mulmuley moves into the top 3. In the case of the query "transaction recovery" C. Mohan's original work on *Aries* is ranked on the top along with Mohan himself; "query optimization" had the entry of Surajit Chaudhari into the top 10 (though was not in the top 10 of the generic "database" query).

5.2.4 Validation of Performance and Scalability

We now discuss some of the experiments that we performed to obtain insights into our algorithm and to validate the system that we built based on the algorithm.

In all these experiments, we consider the average rank of the answers to the experts' queries as a measure of accuracy of the system. One experiment was to determine the effect of the number of terms computed in (6) on the average ranks of these golden answers. The number of terms used to approximate $\mathbf{B}$ is in fact the number of iterations in our algorithm. Figures 3, 4 and 5 plots the average rank versus the number of terms used to compute the $\mathbf{B}$ for the different datasets. Here, the number of terms = 0 stands for the identity matrix and so on.

We find that the average rank starts to decrease with iterations and stabilizes after 6 or 7 iterations in the case of Intranet. The DBLP data is quite shallow and has a lot of dead-end nodes (mainly author nodes), hence we see that average rank stabilizes after 4 to 5 iterations. The Web

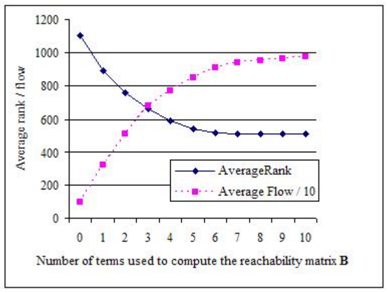

Figure 3: Intranet: Average rank plotted across number of terms used to compute the B.

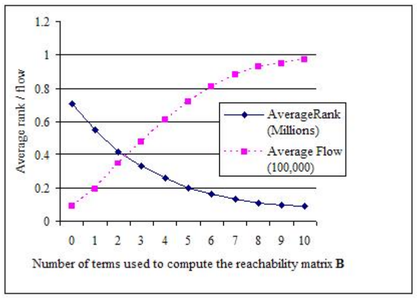

Figure 4: WWW: Average rank plotted across number of terms used to compute the B.

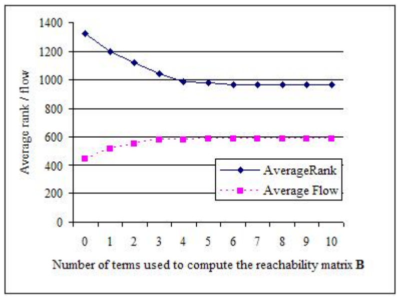

Figure 5: DBLP: Average rank plotted across number of terms used to compute the B.

data is quite dense and well connected (Average density of the **B** matrix is 90) and takes 10 to 11 iterations to stabilize. Another observation of this plot further strengthens the premise that anchor text helps in searching [1]. The average rank when number of terms = 0 (i.e, when the identity matrix is used) is just a ranking based on the document's in-link label distribution and this by itself does surprisingly well.

Figures 3, 4 and 5 also plots the total flow of the system versus the number of terms used to compute **B**. We can see that the average rank saturates much before the total flow saturates. This indicates that the rank of important pages stabilizes fairly early and does not change.

The next obvious question that arises is the performance of the system. As previously mentioned in Section 3, we had a few heuristics to speed up the computations. One of them is to consider only the top m reachable nodes in **B** for every node i while computing the sparse matrix multiplication with $\mathbf{v}_q$. An experiment that helped us decide on an optimal m was to measure the average rank of the same golden answers across various values of m. This is plotted in Figures 6, 7 and 8 where, $m = 0$ is the ranking based on just in-link distribution of labels.

We also plot the average time taken to answer these queries across various values of m in the figures 9, 10 and 11. The time is total time taken (Java's System.currentTimeMillis()) to compute the ranks given a query.

Figures 9, 10 and 11 shows that for $m = 10$, the average time taken to answer queries is hardly 0.8 seconds for intranet, negligible for the dblp dataset and around 2 seconds for the Internet data, using a Java implementation. The time taken to answer the query increases with m, the number reachable nodes. This is because the sparse vector and sparse matrix multiplication tends towards $O(n^2)$ as number of elements in the matrix increases. But, the saving grace is that, figures 6, 7 and 8 show that for low values of m the average rank is low, making this system viable. A notable change in trend is in the dblp data set (time taken to answer a query becomes constant after $m = 100$). This is due to the fact that it is a sparse data-set with few links and low reachability.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we introduced a flow-based model and an efficient method to dynamically compute a ranking of documents for any given label in the anchor text. We established a relationship between our method and traditional random surfer models. We analyzed the convergence behavior of our algorithm on arbitrary graphs, and provided the structural conditions that G must satisfy for our algorithm to produce a label-biased ranking that is different from traditional *PageRank*. Our method can be applied to the Web-graph to obtain a ranking of hyperlinked documents biased by labels in the anchor text. Our model is particularly well suited for fine-grained ranking and does not use a classifier at any step of the ranking process.

We built a system based on our method and conducted experiments on three datasets. Our method gave average rank lower that the average rank given by static *PageRank* by more than an order of 1,000. Our method also gave significantly better results than context-sensitive (or topic-sensitive) *PageRank* at fine-grained topic levels. These results were obtained while keeping the average time to answer a query quite low.

As future work, one can use thesauri to improve the performance of our method at broad-topic levels. It will be interesting to apply our model and extend our method to classify pages, do focused advertising and detect "buzz"/hot topics on the Web.

7. ACKNOWLEDGMENTS

We would like to express our gratitude to Raghu Krishnapuram, Sridhar Rajagopalan and Andrew Tomkins for their time and valuable discussions. We are also indebted to Sridhar, Marcus Fontoura and their team-mates for giving us access to the Intranet data of IBM.

8. REFERENCES

[1] E. Amitay, D. Carmel, A. Darlow, M. Herscovici, A. S. L. Lempel, R. Kraft, and J. Zien. Juru at trec 2003 - topic distillation using query-sensitive tuning and cohesiveness filtering. In *Proceedings of The Twelfth Text Retrieval Conference (TREC 2003)*, Gaithersburg, Maryland, USA, 2003.

[2] G. Bhalotia, C. Nakhe, A. Hulgeri, S. Chakrabarti, and S. Sudarshan. Keyword searching and browsing in databases using BANKS. In *ICDE*, 2002.

[3] K. Bharat and M. R. Henzinger. Improved algorithms for topic distillation in a hyperlinked environment. In *Proceedings of SIGIR-98, 21st ACM International*

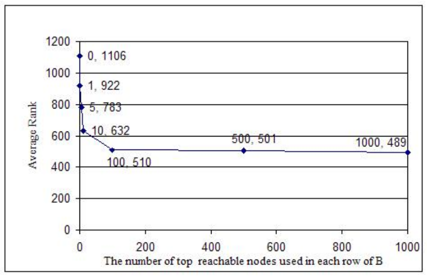

Figure 6: Intranet: Average rank plotted across top m highest reachable nodes from each row of B matrix .

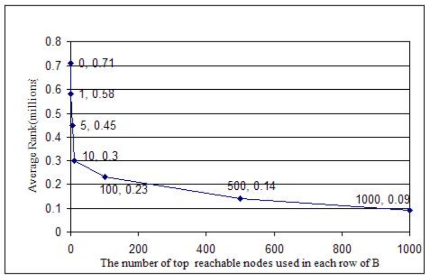

Figure 7: WWW: Average rank plotted across top m highest reachable nodes from each row of B matrix .

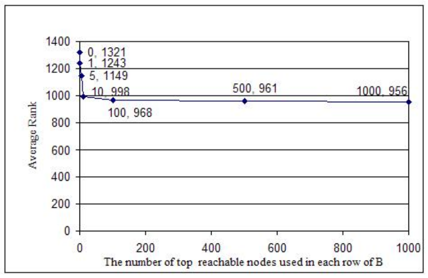

Figure 8: DBLP: Average rank plotted across top m highest reachable nodes from each row of B matrix .

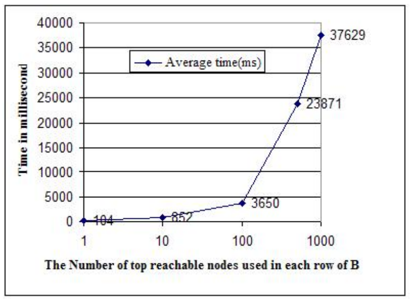

Figure 9: Intranet: Average time taken (ms) to answer a query considering m reachable nodes .

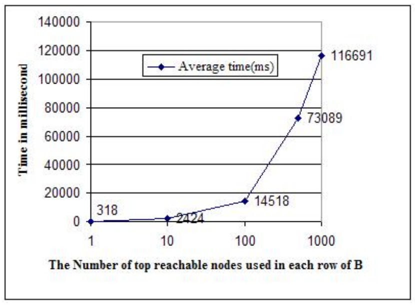

Figure 10: WWW: Average time taken (ms) to answer a query considering m reachable nodes .

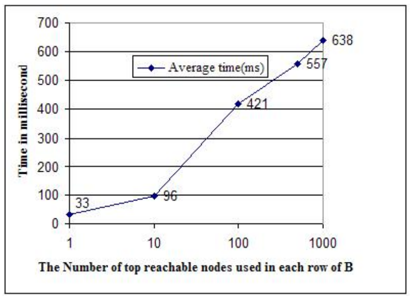

Figure 11: DBLP: Average time taken (ms) to answer a query considering m reachable nodes .

Conference on Research and Development in Information Retrieval, pages 104–111, Melbourne, AU, 1998.

[4] P. Bonacich. Power and centrality: A family of measures. *American Journal of Sociology*, 92, 1987.

[5] S. Brin, R. Motwani, L. Page, and T. Winograd. What can you do with a web in your pocket? *Data Engineering Bulletin*, 21(2):37–47, 1998.

[6] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, and R. Stata. Graph structure in the web. In *In Proceedings of the 9th International World Wide Web Conference*, pages 247–256, 2000.

[7] S. Chakrabarti, B. Dom, D. Gibson, J. Kleinberg, P. Raghavan, and S. Rajagopalan. Automatic resource list compilation by analyzing hyperlink structure and associated text. In *Proceedings of the 7th International World Wide Web Conference*, 1998.

[8] S. Chakrabarti, B. E. Dom, S. R. Kumar, P. Raghavan, S. Rajagopalan, A. Tomkins, D. Gibson, and J. Kleinberg. Mining the Web's link structure. *Computer*, 32(8):60–67, 1999.

[9] N. E. Friedkin. Theoretical foundations for centrality measures. *American Journal of Sociology*, 96(6):1478–1504, 1991.

[10] T. Haveliwala. Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search. *IEEE Transactions on Knowledge and Data Engineering*, July/Aug 2003.

[11] M. R. Henzinger. Hyperlink analysis for the web. *IEEE Internet Computing*, pages 45–50, 2001.

[12] C. Hubbell. An input-output approach to clique identification. *Sociometry*, 28, 1965.

[13] L. Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18, 1953.

[14] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[15] M. Marchiori. The quest for correct information on the web: Hyper search engines. In *Proceedings of the 6th International World Wide Web Conference*, Santa Clara, April 1997.

[16] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.

[17] G. Pinski and F. Narin. Citation influence for journal aggregates of scientific publications: Theory, with application to the literature of physics. *Information Processing and Management*, 12:297–312, 1976.

[18] D. Rafiei and A. O. Mendelzon. What is this page known for? computing web page reputations. In *In Proceedings of the Ninth International World Wide Web Conference*, 2000.

[19] J. A. Tomlin. A new paradigm for ranking pages on the world wide web. In *Proceedings of The Twelfth International World Wide Web Conference*, 2003.