Semi-Supervised Ranking on Very Large Graphs with Rich Metadata

Bin Gao
Microsoft Research Asia
No. 5, Danling Street
Beijing, 100080, P. R. China
bingao@microsoft.com

Tie-Yan Liu
Microsoft Research Asia
No. 5, Danling Street
Beijing, 100080, P. R. China
tyliu@microsoft.com

Wei Wei*
School of Computer Science and Technology
Huazhong University of Science and Technology
Wuhan, 430074, P. R. China
weiwei8329@gmail.com

Taifeng Wang
Microsoft Research Asia
No. 5, Danling Street
Beijing, 100080, P. R. China
taifengw@microsoft.com

Hang Li
Microsoft Research Asia
No. 5, Danling Street
Beijing, 100080, P. R. China
hangli@microsoft.com

ABSTRACT

Graph ranking plays an important role in many applications, such as page ranking on web graphs and entity ranking on social networks. In applications, besides graph structure, rich information on nodes and edges and explicit or implicit human supervision are often available. In contrast, conventional algorithms (e.g., PageRank and HITS) compute ranking scores by only resorting to graph structure information. A natural question arises here, that is, how to *effectively and efficiently* leverage all the information to more accurately calculate graph ranking scores than the conventional algorithms, assuming that the graph is also very large. Previous work only partially tackled the problem, and the proposed solutions are also not satisfying. This paper addresses the problem and proposes a general framework as well as an efficient algorithm for graph ranking. Specifically, we define a semi-supervised learning framework for ranking of nodes on a very large graph and derive within our proposed framework an efficient algorithm called Semi-Supervised PageRank. In the algorithm, the objective function is defined based upon a Markov random walk on the graph. The transition probability and the reset probability of the Markov model are defined as parametric models based on features on nodes and edges. By minimizing the objective function, subject to a number of constraints derived from supervision information, we simultaneously learn the optimal parameters of the model and the optimal ranking scores of the nodes. Finally, we show that it is possible to make the algorithm efficient to handle a billion-node graph by taking advantage of the sparsity of the graph and implement it in the MapReduce logic. Experiments on real data from a commercial search engine show that the proposed algorithm can outperform previous algorithms on several tasks.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; H.5.4 [**Information Interface and Presentation**]: Hypertext/Hypermedia.

General Terms

Algorithm, Experimentation, Theory

Keywords

Page importance, PageRank, MapReduce.

1. INTRODUCTION

Graph ranking is one of the key technologies in applications like web search and social computing. Traditionally, PageRank [7, 17] and HITS [15] algorithms and their variants [13, 16, 19] are exploited in the task, most of which only make use of the link structure information. That is why these algorithms are also known as link structure analysis. Nowadays, extremely large graphs with billions or even trillions of nodes have been created, and a large amount of information has also been accumulated around the graphs. The information is also varied, and includes user behavior data and rich metadata associated with the graphs. For example, in web search, data on users' browsing history of web pages has been collected which can be viewed as humans' supervision on page importance ranking (intuitively, the more visits a page has the more important it is). In addition, web pages contain URLs, contents, anchor texts, tags assigned by users, etc., which can also be useful signals for determining the quality, richness, and freshness of web pages and thus for performing page importance ranking. Similarly, information on edges is helpful as well, such as the type of the hyperlinks and the number of transitions on the hyperlinks.

*This work was performed when the third author was an intern at Microsoft Research Asia.

KDD'11, August 21–24, 2011, San Diego, California, USA.

Thus, graph ranking faces a significant challenge: that is, how to effectively leverage all the useful information to perform better graph ranking and to do it efficiently in order to handle billion-node or trillion-node graphs.

Previous work only partially addressed the challenge. Several approaches have been proposed to incorporate supervision information into the learning process to guide graph ranking [2, 3, 4, 9, 10, 18, 22, 23]. However, these algorithms still have the following issues. (i) Only the graph structure is considered in the algorithms, and some of the useful information on the graph is ignored. (ii) Most algorithms require computations which are expensive in terms of both time and space (nearly $O(n^3)$, where n is the number of nodes), and the algorithms become intractable when the scale of the problem is large.

In this paper, we propose a semi-supervised learning framework for graph ranking to tackle the challenge. In the framework, a graph is no longer considered a simple collection of nodes and edges. Instead, we assign each node and each edge a feature vector, representing meta information about the node and the edge. The ranking model ranks the nodes according to the meta information contained in the graph representation as well as the information from human supervision. Ranking of nodes and learning of model parameters are conducted at the same time and in a theoretically sound way, by minimizing a graph-based objective function with regard to parameterized models of the meta information, subject to constraints from the supervision information.

In addition, we propose an efficient algorithm called *Semi-Supervised PageRank* (SSP), within the framework. Specifically, we (i) define the graph-based objective function using a Markov random walk (similarly as in PageRank); (ii) define the transition probability of the Markov process using a parametric model containing features on edges, and define the reset probability using a parametric model containing features on nodes; (iii) define the constraints as pairwise preferences from supervision information; and (iv) obtain the optimal parameters and optimal ranking result simultaneously by minimizing the graph-based objective function subject to the constraints. Furthermore, by leveraging the fact that graphs are usually sparse, we implement the optimization process in nearly linear complexity with regard to the number of edges. We also show how to parallelize the algorithm using the MapReduce logic [12].

We have conducted experiments on different ranking tasks and with different graph data sets (up to billions of nodes) to test the performance of the proposed algorithm. The experimental results show that our proposed SSP algorithm can outperform existing algorithms on anti-spam and relevance ranking in search. Furthermore, the proposed algorithm can handle billion-scale graphs while most other algorithms cannot. These results demonstrate the advantages of the proposed algorithm.

To sum up, the contributions of the paper are as below. (i) We have proposed a novel learning problem, i.e., ranking on a very large graph with rich metadata, and developed a general framework to address the problem. (ii) We have developed the SSP algorithm, which has very good ranking performance and can scale up to rank billions of nodes. (iii) We have performed an empirical study on state of the art page ranking algorithms using billion-scale web graphs, which is, as far as we know, the largest-scale experiment reported in the literature.

The rest of the paper is organized as follows. The proposed semi-supervised learning framework is presented in Section 2. The SSP algorithm is introduced in Section 3. The related work is discussed in Section 4. In Section 5, we show that several related algorithms actually correspond to special cases of the proposed framework. Experimental results are reported in Section 6. Conclusions and future work are given in the last section.

2. SEMI-SUPERVISED LEARNING FRAMEWORK FOR GRAPH RANKING

In this section, we define a general framework for semi-supervised graph ranking, which enables us to develop powerful graph ranking algorithms.

We first propose a new representation of a graph. Traditionally, a graph is defined as a tuple $\mathcal{G}'(V, E, W)$, where V is a node set, E is an edge set, and W is a weight matrix on the edges in E. In this definition, only the skeleton of the graph can be described, and rich information about the nodes and edges cannot be expressed. To tackle the problem, we propose defining a graph using the following new representation, $\mathcal{G}(V, E, X, Y)$. That is, a graph still contains a node set V with n nodes and an edge set E with m edges. In addition, we also have a set of edge features $X = \{x_{ij}\}$ and node features $Y = \{y_i\}$, which can encode information in the graph. More specifically, for each edge from node i to node j, there is an l-dimensional feature vector $x_{ij} = (x_{ij1}, x_{ij2}, \cdots, x_{ijl})^T$; and for each node i, there is an h-dimensional feature vector $y_i = (y_{i1}, y_{i2}, \cdots, y_{ih})^T$. Usually, l and h are small numbers as compared to the scale of the graph.

The edge weight in traditional graph representation can be regarded as an edge feature. In addition, one can have many other edge features. For example, in web search, the intra-site or inter-site property of an edge (hyperlink) can be a valuable edge feature. Similarly, one can have many node features. Again in web search, the quality, content, and freshness of a node (web page) can be useful node features.

To sum up, in the framework, a graph consists of graph structure V and E, edge features X, and node features Y. The graph structure defines the global relationship among nodes, edge features represent the local relationships between any two nodes, and node features describe the properties of individual nodes. They are all useful information for graph ranking.

Next, we define the semi-supervised graph ranking framework as follows,

$$\min_{\omega \geq 0, \phi \geq 0, \pi \geq 0} R(\pi; f(\omega, X), g(\phi, Y)) \quad (1)$$
$$s.t.\ S(\pi; B, \mu) \geq 0.$$

Here, we represent the ranking scores of nodes in graph $\mathcal{G}(V, E, X, Y)$ as an n-dimensional vector π. Furthermore, we introduce parameter vectors ω and ϕ for the node and edge features, and the corresponding functions $f(\omega, X)$ and $g(\phi, Y)$. The constraints on ω, ϕ, and π are $\omega \geq 0$, $\phi \geq 0$, and $\pi \geq 0$. Note that here by using "≥ 0", we mean that all the elements in the vector are non-negative and at least one element is positive. We will explain these components in the following subsections.

2.1 Objective Function

The objective function in the newly-defined framework takes the following form:

$$R(\pi; f(\omega, X), g(\phi, Y)).$$

This objective function can be understood as a graph-based smoothing function for ranking scores π. It ensures that the ranking scores π are consistent with the information contained in the graph in an *unsupervised* learning process. For example, we can use the Markov random walk model to build the smoothing function. The transition probability of the Markov process is defined with $f(\omega, X)$, the parametric model based on edge features, and the reset probability is defined with $g(\phi, Y)$, the parametric model based on node features. Then the smoothing function requires that the ranking scores should be as close to the stationary distribution of the *parametric* Markov process as possible. More details about this example will be introduced in Section 3.

2.2 Constraints

The constraints in the framework take the following form:

$$S(\pi; B, \mu) \geq 0,$$

where matrix B encodes supervision information, and μ denotes weights on samples of supervision information.

The constraints require that the ranking scores π should be consistent with supervision information as much as possible. Learning with regard to the constraints is performed in a *supervised* manner. In contrast, learning with regard to the objective function is performed in an *unsupervised* manner. We therefore call (1) a *semi-supervised* framework for graph ranking.

Matrix B can represent different types of supervision information, such as binary labels, pairwise preferences, partial order, and even total order. For example, pairwise preferences can be labeled by human annotators or mined from users' implicit feedback. In this case, B is a r-by-n matrix with 1, -1, and 0 as elements, where r is the number of preference pairs. Each row of B represents a pairwise preference $u \succ v$, meaning that node u is preferred to node v. The corresponding row of B has 1 in u's column, -1 in v's column, and zeros in the other columns. Accordingly, the constraints can be specified as below, where e is an r-dimensional vector with all its elements equal to 1.

$$S(\pi; B, \mu) = -\mu^T(e - B\pi) \geq 0. \quad (2)$$

For binary labels (e.g., in web search, spam pages and junk pages may correspond to label zero while good pages correspond to label one), partial order, or total order, it is not difficult to convert them to a number of pairwise preference relations and then use constraints similar to (2).

2.3 Equivalent Optimization Problem

For ease of computation, we convert the constraints into a loss function and add it into the objective function (1), obtaining the following equivalent optimization problem.

$$\min_{\omega \geq 0, \phi \geq 0, \pi \geq 0} \alpha R(\pi; f(\omega, X), g(\phi, Y)) - (1-\alpha) S(\pi; B, \mu), \quad (3)$$

where $0 \leq \alpha \leq 1$. By solving Problem (1) or (3), we can obtain the optimal ranking scores π^* as well as the optimal parameters ω^* and ϕ^*.

3. SEMI-SUPERVISED PAGERANK

In this section, we propose an efficient algorithm named *Semi-Supervised PageRank* (SSP) under the general framework. This algorithm demonstrates, as a showcase, that we can successfully address the challenge to graph ranking presented in Section 1.

3.1 Objective Function and Constraints

In SSP, we define the objective function using the same principle as in PageRank. Specifically, one step of the Markov random walk in PageRank can be written as below,

$$\tilde{\pi} = dP_0^T \pi + (1-d) r_0, \quad (4)$$

where P_0 is the transition matrix, r_0 is the reset probability, and d is the damping factor. We introduce parameters to both the transition matrix and the damping factor, and define the loss of $\|\tilde{\pi} - \pi\|^2$ as the objective function,

$$R(\pi; f(\omega, X), g(\phi, Y)) = \|df(\omega, X)\pi + (1-d)g(\phi, Y) - \pi\|^2. \quad (5)$$

Here $f(\omega, X)$ plays the same role as P_0^T does in the original PageRank algorithm (4). For ease of understanding, we rewrite it as $P^T(\omega, X)$ (or sometimes $P^T(\omega)$ or P^T for compact reading) to show that it is a parametric transition probability matrix. Each element $p_{ij}(\omega)$ in $P(\omega)$ represents the transition probability from node i to node j, which is determined by the model of edge features with parameter ω. For example, we can use linear combination, i.e.,

$$p_{ij}(\omega) = \begin{cases} \frac{\sum_k \omega_k x_{ijk}}{\sum_j \sum_k \omega_k x_{ijk}}, & \text{if there is an edge from } i \text{ to } j, \\ 0, & \text{otherwise.} \end{cases} \quad (6)$$

That is, only the transition probability for an existing edge in the graph is non-zero[1], and the value is determined by the edge features. We will only change the weight of an existing edge including assigning zero weight, but will not add new edges to the graph. This can keep the graph sparse.

Here $g(\phi, Y)$ plays the same role as r_0 in the original PageRank algorithm (4). We also rewrite it as $r(\phi, Y)$ (or sometimes $r(\phi)$ or r for compact reading) to show that it is a parametric reset probability vector. Each element $r_i(\phi)$ in $r(\phi)$ represents the reset probability of node i, which is determined by the model of node features with parameter ϕ. For example, we can once again use linear combination, i.e.,

$$r_i(\phi) = \phi^T y_i. \quad (7)$$

Suppose the constraints are based on pairwise preferences.[2] We then obtain the following optimization problem.

$$\min_{\omega \geq 0, \phi \geq 0, \pi \geq 0} \{\alpha \|dP^T(\omega)\pi + (1-d)r(\phi) - \pi\|^2 + (1-\alpha)\mu^T(e - B\pi)\}. \quad (8)$$

One may think about whether it is possible to move the constraints $\omega \geq 0$, $\phi \geq 0$, and $\pi \geq 0$ into the objective function and formalize the problem as non-constrained optimization. It is not an appropriate formulation, however. This is because the features and ranking scores must be non-negative in the current problem and the three constraints

[1]To avoid rank sink, if a node has no outlink edges, we will assume it links to all nodes.

[2]As mentioned before, all kinds of supervision information can be converted to pairwise preferences.

must be strictly satisfied. The non-negative conditions will not be guaranteed when included in the objective function. In fact, in our formulation we need to conduct boundary detection for ω, ϕ, π in each iteration to ensure that they are non-negative. By solving this problem[3], we can obtain the optimal ω^*, ϕ^*, and π^*, and use them to rank the nodes in the graph. In the next sub-sections, we will show how to efficiently solve this optimization problem.

3.2 Solving the Optimization Problem

For ease of explanation, we use $G(\omega, \phi, \pi)$ to denote the objective function in (8),

$$G(\omega, \phi, \pi) = \alpha \|dP^T(\omega)\pi + (1-d)r(\phi) - \pi\|^2 + (1-\alpha)\mu^T(e - B\pi). \quad (9)$$

We use the gradient descent method to minimize $G(\omega, \phi, \pi)$. The partial derivatives of $G(\omega, \phi, \pi)$ with respect to ω, ϕ, and π can be calculated as below,

$$\frac{\partial G}{\partial \omega} = 2\alpha d[P^T\pi \otimes \pi - \pi \otimes \pi + (1-d)r \otimes \pi]^T \frac{\partial vec(P)}{\partial \omega^T}, \quad (10)$$

$$\frac{\partial G}{\partial \phi} = 2\alpha(1-d)[(1-d)r + dP^T\pi - \pi]\frac{\partial r}{\partial \phi}, \quad (11)$$

$$\frac{\partial G}{\partial \pi} = 2\alpha[(dPP^T - dP - dP^T + I)\pi - (1-d)(I - dP)r] - (1-\alpha)B^T\mu. \quad (12)$$

Operator $\otimes$ represents the Kronecker product, and operator $vec(\cdot)$ denotes the expansion of a matrix to a long vector by its columns. For the last fractions in (10) and (11), we have,

$$\frac{\partial vec(P)}{\partial \omega^T} = \begin{pmatrix} \frac{\partial p_{11}}{\partial \omega_1} & \cdots & \frac{\partial p_{11}}{\partial \omega_l} \\ \vdots & \ddots & \vdots \\ \frac{\partial p_{n1}}{\partial \omega_1} & \cdots & \frac{\partial p_{n1}}{\partial \omega_l} \\ \vdots & \ddots & \vdots \\ \frac{\partial p_{1n}}{\partial \omega_1} & \cdots & \frac{\partial p_{1n}}{\partial \omega_l} \\ \vdots & \ddots & \vdots \\ \frac{\partial p_{nn}}{\partial \omega_1} & \cdots & \frac{\partial p_{nn}}{\partial \omega_l} \end{pmatrix} \text{ and } \frac{\partial r}{\partial \phi} = \begin{pmatrix} \frac{\partial r}{\partial \phi_1} \\ \vdots \\ \frac{\partial r}{\partial \phi_i} \\ \vdots \\ \frac{\partial r}{\partial \phi_h} \end{pmatrix} \quad (13)$$

If $p_{ij}(\omega)$ is a linear function of the edge features, its partial derivatives with respect to ω_k will be,

$$\frac{\partial p_{ij}}{\partial \omega_k} = \frac{x_{ijk}\sum_j\sum_k \omega_k x_{ijk} - (\sum_k \omega_k x_{ijk})(\sum_j x_{ijk})}{(\sum_j \sum_k \omega_k x_{ijk})^2}. \quad (14)$$

With the above derivatives, we can iteratively update ω, ϕ, and π by means of gradient descent. The corresponding algorithm is given in Figure 1, where ρ is the learning rate and ϵ controls the stopping condition. Note that the problem does not have a unique minimum. We may compute the original PageRank for the graph and use it as the initial value of π.

[3]Note that we do not add $\sum_i \pi_i = 1$ as a constraint to the optimization problem. In our mind, though this constraint looks necessary if π is supposed to be a probability distribution in PageRank, it is not so necessary if we use it for ranking, where the absolute value of π_i is not very important as long as the relative order is preserved.

Input: X, Y, B, μ, l, h, n, ρ, ϵ, α.
Output: Node ranking score π^*

Algorithm:

1. Set $s = 0$, initialize $\pi_i^{(0)}$ $(i = 1, \cdots, n)$, $\omega_k^{(0)}$ $(k = 1, \cdots, l)$, and $\phi_t^{(0)}$ $(t = 1, \cdots, h)$.
2. Calculate $P^{(s)} = P(\omega^{(s)})$, $r^{(s)} = r(\phi^{(s)})$, and $G^{(s)} = G(\omega^{(s)}, \phi^{(s)}, \pi^{(s)})$.
3. Update $\pi_i^{(s+1)} = \pi_i^{(s)} + \rho\frac{\partial G^{(s)}}{\partial \pi_i^{(s)}}$, $\omega_k^{(s+1)} = \omega_k^{(s)} + \rho\frac{\partial G^{(s)}}{\partial \omega_k^{(s)}}$, and $\phi_t^{(s+1)} = \phi_t^{(s)} + \rho\frac{\partial G^{(s)}}{\partial \phi_t^{(s)}}$.
4. Force $\pi_i^{(s+1)}$, $\omega_k^{(s+1)}$, $\phi_t^{(s+1)}$ to 0, if they are negative.
5. Normalize $\pi_i^{(s+1)} \leftarrow \frac{\pi_i^{(s+1)}}{\sum_{j=1}^n \pi_j^{(s+1)}}$, $\omega_k^{(s+1)} \leftarrow \frac{\omega_k^{(s+1)}}{\sum_{j=1}^l \omega_j^{(s+1)}}$, and $\phi_t^{(s+1)} \leftarrow \frac{\phi_t^{(s+1)}}{\sum_{j=1}^h \phi_j^{(s+1)}}$.
6. Calculate $G^{(s+1)} = G(\omega^{(s+1)}, \phi^{(s+1)}, \pi^{(s+1)})$, if $G^{(s)} - G^{(s+1)} < \epsilon$, stop and output $\pi^* = \pi^{(s+1)}$; else $s = s + 1$, jump to step 2.

Figure 1: The Learning Process of the SSP Algorithm.

3.3 Efficient Implementation

Next, we efficiently implement the algorithm, by using the common fact that graphs in practice tend to be sparse.

By defining $\pi' = P^T\pi$ and $\pi'' = \pi' - \pi$, and conducting some simple mathematical transformations, we can reduce the partial derivative on π to,

$$\frac{\partial G}{\partial \pi} = 2\alpha[d(P\pi'' - \pi'') + (1-d)(\pi - r + dPr)] - (1-\alpha)B^T\mu. \quad (15)$$

In order to compute (15), only three steps of matrix-vector multiplications are needed: $P^T\pi$, $P\pi''$, and Pr. Similarly, the computations in (10) and (11) can also be simplified with the help of π' and π'', i.e.,

$$\frac{\partial G}{\partial \omega} = 2\alpha d\{[\pi'' + (1-d)r] \otimes \pi\}^T \frac{\partial vec(P)}{\partial \omega^T}. \quad (16)$$

$$\frac{\partial G}{\partial \phi} = 2\alpha(1-d)[(1-d)r + d\pi' - \pi]\frac{\partial r}{\partial \phi}. \quad (17)$$

Since the graph is sparse, in (16), we only need to compute the non-zero blocks in the Kronecker product and the left partial derivative vector in (13). Suppose there are m edges in the graph, then the cost is proportional to m. In general, the computational complexity of the proposed Semi-Supervised PageRank algorithm is only of order $O(ml + n)$.

Moreover, we can parallelize the algorithm using MapReduce. MapReduce [12] is a programming model for parallelizing large-scale computations on a distributed computer cluster. It reformulates the logic of a computation task to a series of Map and Reduce operations. The map operation takes a <key, value> pair, and emits one or more intermediate <key, value> pairs. Then all values with the same intermediate key are grouped together into a <key, valuelist> pair, where valuelist contains all values associated with the same key. The reduce operation reads a <key, valuelist> pair and emits one or more new <key, value> pairs.

There are mainly two kinds of large-scale computation prototypes in SSP, i.e., matrix-vector multiplication and Kronecker product of vectors on a sparse graph. These proto-

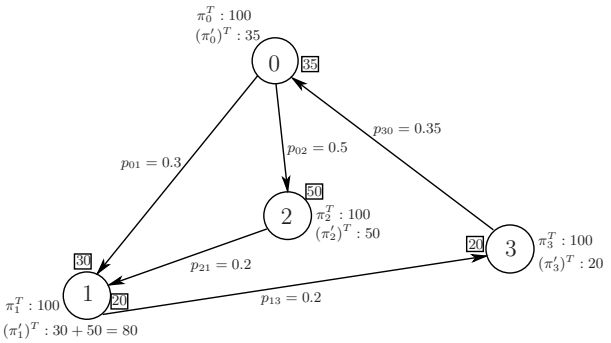

Figure 2: Matrix P is built from the graph, in which element p_{ij} is defined as the normalized edge weight from node i to node j. The product $\pi^T P$ can be interpreted as the data propagation process on the graph. $\pi^T \Rightarrow (\pi')^T$.

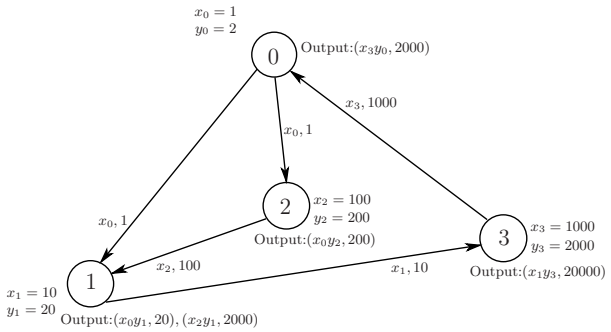

Figure 3: Kronecker product of two vectors is trying to compute $x_i y_j$ under the constraint of graph structure. We can propagate x_i along graph edges, so as to filter necessary x_i to multiply y_j. The process can be implemented by two MapReduce processes.

types can be written in the below MapReduce logic, in which we denote O_i as the outlink node set of node i, and I_i as the inlink node set of node i.

3.3.1 Matrix-Vector Multiplication

We take $\pi' = P^T \pi$ as example. It can be equally written as $(\pi')^T = \pi^T P$. The computation can be performed using a graph based propagation process. That is, we first take π^T as a meta data of node i, and then pass π^T from node i to each of its connected node j by multiplying p_{ij}. By aggregating all the incoming values on each node j, we can get the new $(\pi')^T$. The process can be implemented as below (refer to Figure 2).

Map: Take graph record $< i, \{j, p_{ij}\}, j \in O_i >$ and $< i, \pi_i >$ as input. Map input on j, such that tuples with the same j are shuffled to the same machine in the form of $< j, \{\pi_i p_{ij}\}, \forall j \in O_i >$.

Reduce: Take $< j, \{\pi_i p_{ij}\}, \forall i \in I_j >$ and emit $< j, \pi'_j >$, where $\pi'_j = \sum_{\forall i \in I_j} \pi_i p_{ij}$.

3.3.2 Kronecker Product of Vectors on a Sparse Graph

Suppose x and y are both n-dimensional vectors, we want to compute the Kronecker product $z = x \otimes y$ (z is an n^2-dimensional vector) of them on a sparse graph, i.e., we need to compute $x_i y_j$ only if there is an edge from page i to page j in the graph. We do not want to look up the sparse graph to determine whether we need to compute $x_i y_j$. The solution is to take x_i, y_i as the meta data of node i in the graph, and pass x_i from node i to its connected nodes. After that, we can aggregate the incoming x_i to node j. By multiplying y_j with all x_i received on node j, we can get all necessary $x_i y_j$. The operations can be implemented as below (refer to Figure 3).

Map-I: Take graph record $< i, \{j, p_{ij}\}, j \in O_i >$ and $< i, x_i >$ as input. Map input on j, such that tuples with the same j are shuffled to the same machine in the form of $< j, (i, x_i) >$.

Reduce-I: Take $< j, (i, x_i), \forall i \in I_j >$, and emit $< j, \{(i, x_i), i \in I_j\} >$.

Map-II: Map $< j, \{(i, x_i), i \in I_j\} >$ and $< j, y_j >$ on j, such that tuples with the same j are shuffled to the same machine in the form of $< j, \{y_j, (i, x_i), i \in I_j\} >$.

Reduce-II: Take $< j, \{y_j, (i, x_i), i \in I_j\} >$, and emit $< i, j, x_i y_j >$.

Besides this, there are some other operations in the SSP algorithm that can also be written in the MapReduce logic, including vector normalization, vector addition (and subtraction), and the gradient updating rules. As the implementations are trivial, we will not give unnecessary details.

3.4 Advantages

As has been seen so far, the SSP algorithm has the following advantages. (i) It can naturally leverage all the information useful for graph ranking, including meta information on graphs and supervision information from humans. (ii) It has a nearly linear time complexity with respect to the number of edges, and thus can scale up to very large graphs. (iii) It employs a parametric model, which has a generalization ability. In fact, with our algorithm, the ranking model can be learned from one graph (or one part of a graph) and applied to the other graphs (or the other parts of a graph).

4. RELATED WORK

In the framework we defined in Section 2, graph structure, edge features, and node features are all considered. If only a part of them are used, we obtain several variants of the objective function. Most recent work on graph ranking can be understood as optimizing such special cases, as shown below.[4] It is obvious that these special cases are not as powerful as the general case using the Type (a) objective function in Table 1.

(i) LiftHITS [10] corresponds to optimizing the Type (d) objective function defined based upon the HITS algorithm, subject to constraints of binary labels derived from user feedback. It cannot leverage information carried by node and edge features, neither can it expand to very-large-scale graphs. When 'lifting' the authority of one node, many corresponding sparse rows in the link matrix will become dense. The problem will become more severe when 'lifting' the authority of multiple nodes. As a result, the computation of the eigenvectors of the link matrix will be prohibitively expensive.

(ii) Adaptive PageRank [22] alters the PageRank scores of nodes according to feedback from humans, using an optimization technique. It corresponds to optimizing the Type (d) objective function subject to constraints of the pairwise

[4]We present the detailed derivations in Section 5.

Table 1: Different forms of the objective function.

TYPE	EDGE	NODE	FORM	EXAMPLE METHODS
(a)	$\surd$	$\surd$	$R(\pi; f(\omega, X), g(\phi, Y))$	Semi-Supervised PageRank
(b)	$\surd$		$R(\pi; f(\omega, X))$	
(c)		$\surd$	$R(\pi; g(\phi, Y))$	
(d)			$R(\pi)$	LiftHITS, Adaptive PageRank, NetRank, Laplacian Rank

preferences. It does not use the edge features, and it has high time complexity. This is because it requires computing the inverse of the link matrix during its optimization process. The authors have proposed grouping nodes into clusters, to reduce the complexity. However, to make effective use of human supervision, the number of clusters cannot be too small and thus the issue has not been fundamentally solved.

(iii) NetRank [9, 3] learns the parameters of the Markov random walk on the graph according to supervision information. It corresponds to optimizing the Type (d) objective function subject to the constraints of the pairwise preferences. The method does not leverage the node and edge features; and it is also inefficient, because it needs to compute successive matrix multiplications multiple times at each step of the optimization process.

(iv) Laplacian Rank [24, 4, 18, 23] formulates the supervised graph ranking problem as that of minimizing a combination of an empirical loss and a graph Laplacian based regularization term. It corresponds to optimizing the Type (d) objective function subject to the constraints of the pairwise preferences. In addition to not being able to use the node and edge features, it cannot scale up due to the necessity of computing the pseudo-inverse of the Laplacian matrix in the optimization process.

To sum up, these previous methods cannot make effective use of information on the nodes and edges, and/or cannot scale up to very large graphs.

In addition, there are some other works on classification or link prediction on graphs with metadata. SPA [11] is a graph node classifier based on kernel smoothing trained by semi-supervised learning. It leverages the information from both the labeled nodes and the relationship between labeled and unlabeled nodes. WITCH [1] is a spam page classifier based on graph regularization. It exploits both the graph structure and the web page content to learn a linear classifier using an SVM-like objective function. Supervised Random Walks [5] combines the network structure with node and edge attributes, and uses the attributes to supervise a random walk on the graph in link prediction. All these works are not for graph ranking, and thus it is not appropriate to make a comparison of them with the current work.

5. DISCUSSION

We show that several previous graph ranking algorithms actually correspond to special cases of the proposed framework in (1) or another form (3).

5.1 LiftHITS

Suppose a link matrix M specifies the connectivity between nodes, i.e., $M_{ij} \neq 0$ if there is a link from node i to node j; $M_{ij} = 0$, otherwise. Then the authority scores π can be iteratively computed by $\pi' = M^T M\pi$ in HITS. If we let $S(\pi; B, \mu) = B(\pi' - \pi)$, and use a Type (d) objective function $R(\pi) = \|\pi' - M^T M\pi\|$, then Problem (1) will become

$$\begin{array}{ll} \min_{\pi \geq 0} & \|\pi' - M^T M\pi\| \\ s.t. & B(\pi' - \pi) \geq 0, \end{array} \tag{18}$$

where matrix B specifies the nodes that should be 'lifted" in the next iteration.

The above formulation is equivalent to the LiftHITS method proposed in [10]. In other words, LiftHITS is a special case of the general framework, in which the Type (d) objective function and the constraints of binary labels are used.

5.2 Adaptive PageRank

If we let $S(\pi; B, \mu) = B\pi - \xi$, where $\xi > 0$ is a slack vector, and adopt a Type (d) objective function $R(\pi) \equiv R(\pi(\eta)) = \|\pi(\eta) - \bar{\pi}\|$, where π and $\bar{\pi}$ are computed from the graph structure, then Problem (1) will become

$$\begin{array}{ll} \min_{\pi \geq 0} & \|\pi(\eta) - \bar{\pi}\| \\ s.t. & B\pi \geq \xi,\ \xi \geq 0 \\ & \bar{\pi} = (1-d)(I - dP)^{-1} e \\ & \pi(\eta) = (1-d)(I - dP)^{-1}\eta \\ & \eta \geq 0. \end{array} \tag{19}$$

where P is the transition matrix derived from the graph, d is the damping factor in PageRank, and η is a n-dimensional parameter vector for the nodes in the graph.

The above formulation exactly corresponds to the Adaptive PageRank method proposed in [22]. In other words, Adaptive PageRank is a special case of the general framework, in which the Type (d) objective function and the constraints of the pairwise preferences are used.

5.3 NetRank

Suppose the transition matrix is parameterized by different types of edges, i.e., edge (i, j) belongs to type $t(i, j)$. If type t has an associated weight $w(t)$, the weight of edge (i, j) will be $w(t(i, j))$. Thus one possible formulation of the parametric transition matrix can be written by

$$P_{ij}(t) = \tfrac{w(t(i,j))}{\sum_j w(t(i,j))}. \tag{20}$$

If we set $\alpha = 0$ and let $S(\pi; B, \mu) = -e^T(e - B\pi)$, Problem (3) will become as follows,

$$\begin{array}{ll} \min_{\pi \geq 0} & \sum_{i \prec j} (1 + \pi_i - \pi_j) \\ s.t. & \pi = (P^T(t))^H \pi^{(0)}. \end{array} \tag{21}$$

where $\pi^{(0)}$ is a vector whose elements all equal $\frac{1}{n}$, and H is the iteration number of matrix multiplication.

The above problem corresponds to one formulation of the NetRank method [9]. In other words, NetRank is a special

case of the general framework, in which the Type (d) objective function and the constraints of the pairwise preferences are used.

5.4 Laplacian Rank

If we define the Type (d) objective function using the Laplacian of the graph, i.e., $R(\pi) = \pi^T L \pi$, where

$$L = I - \frac{\Pi^{1/2} P \Pi^{-1/2} + \Pi^{-1/2} P^T \Pi^{1/2}}{2} \tag{22}$$

is the Laplacian for the directed graph, and Π is a diagonal matrix with $\Pi_{ii} = \bar{\pi}_i$ ($\bar{\pi}$ has the same definition as in Section 5.2), use the pairwise preferences as the constraints, and set $\alpha = \frac{1}{2}$, Problem (3) will become,

$$\begin{array}{ll} \min_{\pi \geq 0} & \frac{1}{2}\pi^T L \pi + \frac{1}{2}\sum_{(i,j)} \mu_{ij}\xi_{ij} \\ s.t. & \pi_i - \pi_j \geq 1 - \xi_{ij}, \xi_{ij} \geq 0. \end{array} \tag{23}$$

This is exactly the Laplacian Rank method discussed in [24, 4, 18, 23].[5] In this regard, we say Laplacian Rank is a special case of the general framework, in which the Type (d) objective function and the constraints of the pairwise preferences are used.

6. EXPERIMENTAL RESULTS

In the experiments, we use page importance ranking on web graphs to evaluate the performance of our proposed SSP algorithm.

Two web graphs of different scales are used. The pages on both graphs belong to the ".uk" domain.

- The first graph, denoted as G_1, is the website graph from the Web Spam Challenge [8, 25], which contains 114,529 nodes and 1,836,441 edges. Some websites in the dataset G_1 are labeled as "spam" or "non-spam", and the labeled data is partitioned into a training set (with 222 spam sites and 3,776 non-spam sites) and a test set (with 122 spam sites and 1,933 non-spam sites).

- The second graph, denoted as G_2, is from a commercial web search engine, which contains about 1.4 billion pages and 15.6 billion edges. Together with the graph, a dataset of queries and their associated web pages are also provided. The dataset is partitioned into a training set (with millions of queries collected from the query log in October 2009) and a test set (with about 2000 queries sampled from the query log in November 2009). In the training set, the click-through count of each page is given as the implicit judgment on its relevance to the query. In the test set, each document is manually labeled as "relevant" or "irrelevant."

For each edge in the graphs, we extract ten features, including (1) the type of the edge (i.e., intra-site or inter-site), (2) the inlink number of the source node of the edge, (3) the outlink number of the source node of the edge, (4) the inlink number of the destination node of the edge, (5) the outlink number of the destination node of the edge, (6) the URL depth of the source node of the edge, (7) the URL depth of the destination node of the edge, (8) the URL length of the source node of the edge, (9) the URL length of the destination node of the edge, and (10) the weight (link number) of the edge. For each node, we extract five features, including (1) the inlink number of the node, (2) the outlink number of the node, (3) the number of two-step neighbors, (4) the URL depth of the node, and (5) the URL length of the node.

[5]The work in [18] used an alternate form of Laplacian Rank in label propagation for classification, and used an alternate form of PageRank in graph ranking.

Table 2: Number of spam websites over buckets.

No.	# of Websites	PageRank	AP	SSP
1	150	2	0	0
2	537	2	0	0
3	1257	1	1	0
4	2660	2	8	2
5	4788	4	7	8
6	8344	12	7	9
7	13708	7	16	12
8	20846	13	33	27
9	29008	19	25	42
10	33231	60	25	22

The parameters for our algorithm are set empirically as follows: (i) μ is set to a vector whose elements are all equal to 1; (ii) ρ is set to 0.1; (iii) ϵ is set to 10^{-12}; (iv) α is set to 0.5. In our experiments, the SSP algorithm converges within 30 iterations.

We run the experiments on a cluster of 40 Rackable C2004 servers each with a quad-core CPU and 8 gigabytes memory. We test two baseline algorithms in addition to our algorithm: PageRank and Adaptive PageRank (AP for short). We are not able to test LiftHITS, NetRank, and Laplacian Rank in our computer cluster, because these algorithms are computationally very demanding. Even in a more powerful computing environment, these algorithms could hardly handle a dataset like G_1 or G_2. In the experiments, the damping factors for SSP, PageRank, and AP are all heuristically set to 0.85. For AP, we conduct clustering on the nodes, with each cluster containing about 10,000 nodes, following the proposal in [22]. Even with clustering, AP can only run on G_1. Therefore we only report the result of AP on G_1. Note that we do not consider the efficient implementations of personalized PageRank [14] such as those in [21] as baselines. The reason is that such personalized PageRank methods largely depend on the preference assumption, like the topic bias in topic-sensitive PageRank[13], which does not hold in the general setting of SSP.

6.1 Anti-Spam

We use the spam labels in the training set of G_1 as supervision information to train different page importance ranking methods, and evaluate the ranking results on the test set to see whether the methods are good at anti-spam.

In order to train SSP and AP, we create pairwise preferences from the spam labels (i.e., non-spam websites are preferred to spam websites). We evaluate the ranking performance using spam bucket distribution. Specifically, given an algorithm, we sort all the websites in G_1 in the descending order of their ranking scores given by the algorithm. Then we put these sorted websites into 10 buckets.[6] The numbers of the labeled spam websites in the test set over the buckets

[6]The buckets are partitioned based on the scores given by the algorithm. That is, the sum of the scores in each bucket is 10% of the total score.

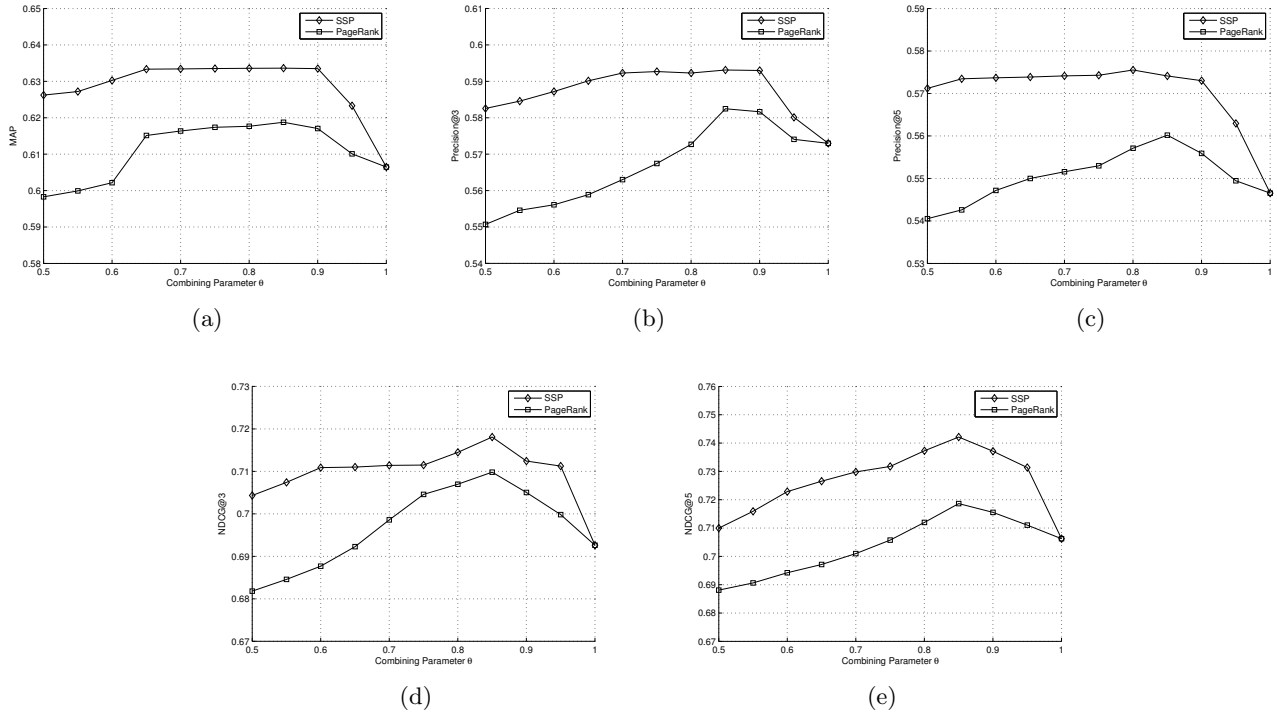

Figure 4: (a) Search performance in terms of MAP. (b) Search performance in terms of Precision@3. (c) Search performance in terms of Precision@5. (d) Search performance in terms of NDCG@3. (e) Search performance in terms of NDCG@5.

for different algorithms are listed in Table 2. Considering that the goal of spammers is to promote the ranking positions of their websites, a ranking algorithm with good anti-spam ability should remove as many spam sites from the top buckets as possible. For the websites in the bottom buckets, as their page importance scores are very low, whether they are spam or not does not make much difference.

From the table, we have the following observations: (i) AP performs better than PageRank, for AP can take advantage of supervision information while PageRank is an unsupervised algorithm. (ii) SSP outperforms AP, for SSP can leverage more information than AP. Specifically, SSP integrates graph structure, edge features, node features, and supervision information into the ranking mechanism, while AP does not consider using edge features and node features. Therefore, SSP can successfully remove more spam websites from the top buckets, compared to the two baselines.

6.2 Relevance Ranking

In web search, the retrieved pages for a given query are often ranked according to two factors: content relevance and page importance. Without loss of generality, we use a linear combination of these two factors to produce the final ranking result,

$$\theta Score_{relevance} + (1-\theta) Score_{importance}, \qquad (24)$$

where $0 \leq \theta \leq 1$ is the combining parameter. In our experiments, we use BM25 [20] to generate the relevance score, and the importance score is calculated by different algorithms under investigation. The corresponding relevance ranking performances are evaluated in terms of MAP [6], Precision@k [6], and NDCG@k [6] on the test set.

As mentioned above, since G_2 is extremely large, only PageRank and SSP can run on the data. For SSP, we construct pairwise preferences from the click-through counts. That is, if the number of clicks of a page for all the queries in the training data is larger than that of the other page, we will create a pairwise constraint on the two pages. PageRank does not use any training data since it is an unsupervised algorithm. After a ranking model is trained by an algorithm, we compute the importance scores of all the pages in G_2 using the ranking model. We then combine the importance score and relevance score to rank the pages in the test set.

MAP of the ranking results are shown in Figure 4 (a). Precision @$3, 5$ of the ranking results are shown in Figure 4 (b) and (c). NDCG@$3, 5$ of the ranking results are shown in Figure 4 (d) and (e). From the figures, we can see that SSP consistently outperforms PageRank, with all θ values, and in terms of all evaluation measures. The explanation for the better performance of SSP is as follows. By leveraging more helpful information (e.g., node and edge features, supervision information), SSP can achieve better performance than PageRank in page importance ranking.

To summarize the experimental results, SSP outperforms the baseline algorithms on several tasks and in several evaluation measures, and can handle billion-scale graphs very well.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we have defined a semi-supervised learning framework for graph ranking on a very-large-scale graph, and developed an efficient algorithm named Semi-Supervised PageRank (SSP) within the framework. The new algorithm can make effective use of supervision information, leverage

rich information on the graph, and scale up to very-large-scale problems. Our experimental results have shown that the proposed algorithm can outperform baseline algorithms on graph ranking.

For future work, we plan to investigate the following issues. (i) We will try advanced optimization techniques like conjugate gradient method to minimize the objective function in Semi-Supervised PageRank, instead of using the simple gradient descent method. (ii) We plan to try other forms of objective functions, so as to extend the idea of SSP to other algorithms such as Semi-Supervised HITS. (iii) We will consider improving SSP using both labeled and unlabeled data as in conventional semi-supervised learning. Currently, the preference matrix B only contains known preference pairs; we may extent B to cover all pairs (both known pairs and unknown pairs) and also predict the relationship of the unknown pairs in the learning process. (iv) We will conduct more comprehensive studies to investigate the sensitivity of the parameters in the model.

8. REFERENCES

[1] J. Abernethy, O. Chapelle, and C. Castillo. Web spam identification through content and hyperlinks. In the proceedings of AIRWeb'08, 2008.

[2] A. Agarwal and S. Chakrabarti. Learning random walks to rank nodes in graphs. In the proceedings of the 24th International Conference on Machine Learning (*ICML*), pages 9-16, 2007.

[3] A. Agarwal, S. Chakrabarti, and S. Aggarwal. Learning to rank networked entites. In the proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 14-23, 2006.

[4] S. Agarwal. Ranking on graph data. In the proceedings of the 23th International Conference on Machine Learning (*ICML*), pages 25-32, 2006.

[5] L. Backstrom and J. Leskovec. Supervised Random Walks: Predicting and Recommending Links in Social Networks. In the proceedings of the 4th ACM International Conference on Web Search and Data Mining (*WSDM*), pages 635-644, 2011.

[6] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, ISBN-13: 978-0201398298, May 1999.

[7] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. In *Computer Networks and ISDN Systems*, volume 30, issue 1-7, pages 107-117, 1998.

[8] C. Castillo, D. Donato, L. Becchetti, P. Boldi, S. Leonardi, M. Santini, and S. Vigna. A reference collection for web spam. In *SIGIR Forum*, volumn 40, issue 2, pages 11-24, 2006.

[9] S. Chakrabarti and A. Agarwal. Learning parameters in entity relationship graphs from ranking preferences. In the proceedings of the 10th European Conference on Principles and Practice of Knowledge Discovery in Databases (*PKDD*), volume 4213, pages 91-102, 2006.

[10] H. Chang, D. Cohn, and A. K. McCallum. Learning to create customized authority lists. In the proceedings of the 17th International Conference on Machine Learning (*ICML*), pages 127-134, 2000.

[11] M. Culp and G. Michailidis. Graph-based semisupervised learning. IEEE Transactions on Pattern Analysis and Machine Intelligence, VOL. 30, No. 1, pages 174-179, 2008.

[12] J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In the proceedings of the 6th Symposium on Operating System Design and Implementation (*OSDI*), 2004.

[13] T. H. Haveliwala. Topic-sensitive PageRank. In the proceedings of the 11th International World Wide Web Conference (*WWW*), 2002.

[14] T. Haveliwala, S. D. Kamvar, and G. Jeh. An Analytical Comparison of Approaches to Personalizing PageRank. Stanford University, Preprint, 2003.

[15] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. In the proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms (*SODA*), pages 668-677, 1998.

[16] T. Kolda and B. Bader. The TOPHITS Model for Higher-Order Web Link Analysis. In the proceedings of the 4th Workshop on Link Analysis, Counterterrorism and Security, in conjunction with the 6th SIAM International Conference on Data Mining (*SDM*), 2006.

[17] 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.

[18] D. Rao, and D. Yarowsky. Ranking and semi-supervised classification on large scale graphs using map-reduce. In the proceedings of the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (*ACL-IJCNLP*), 2009.

[19] M. Richardson and P. Domingos. The intelligent surfer: probabilistic combination of link and content information in PageRank. In the proceedings of the 16th Annual Conference on Neural Information Processing Systems (*NIPS*), 2002.

[20] S. E. Robertson. Overview of okapi projects. *Journal of Documentatioin*, volumn 53, issue 1, pages 3-7, 1997.

[21] T. Sarlos, A. A. Benczur, K. Csalogany, D. Fogaras, and B. Racz. To randomize or not to randomize: space optimal summaries for hyperlink analysis. In the proceedings of the 15th International World Wide Web Conference (*WWW*), pages 297-306, 2006.

[22] A. C. Tsoi, G. Morini, F. Scarselli, M. Hagenbuchner, and M. Maggini. Adaptive ranking of Web pages. In the proceedings of the 12th International World Wide Web Conference (*WWW*), pages 356-365, 2003.

[23] M. Xie, J. Liu, N. Zheng, D. Li, Y. Huang, and Y. Wang. Semi-supervised graph-ranking for text retrieval. In the proceedings of the 4th Asia Infomation Retrieval Symposium (*AIRS*), pages 256-263, 2008.

[24] D. Zhou, J. Huang, and B. Scholkopf. Learning from labeled and unlabeled data on a directed graph. In the proceedings of the 22th International Conference on Machine Learning (*ICML*), pages 1041-1048, 2005.

[25] http://www.yr-bcn.es/webspam/datasets/uk2007/