A New Approach for Ranking Web Pages

Azadeh Shakery Mohammad Ghodsi

shakeri@ce.sharif.edu ghodsi@sina.sharif.edu

Computer Engineering Department
Sharif University of Technology

Abstract

Today, information retrieval is a common task for internet users. The huge size and heterogeneous structure of the web has caused new problems for its users. In this immense resource of information, it is exceedingly difficult for users to locate resources that are both high in quality and relevant to their information needs. One of the main tasks of each search engine is providing users with these high quality and relevant resources. Search engines use ranking algorithms in order to sort the documents related to the user need.

For ranking web resources, several ranking methods have been introduced. In this paper, we present the most famous web ranking algorithms and briefly explain their advantages and drawbacks. We also propose a new algorithm for ranking web pages, which exploits hyperlink structure as well as document contents for ranking resources, with the intent of overcoming the drawbacks of current ranking algorithms.

Keywords: *Ranking, PageRank, Kleinberg Algorithm, Information Retrieval*

1 Introduction

Web is a set of unstructured information that is expanding with a high rate. Several studies have estimated the size of the web, and while they report slightly different numbers, most of them agree that there are more than a billion pages on the web.

In this huge and unstructured set of information, search engines face different difficulties. The search engines receive queries and return a set of web pages as the result. If the query posed to the search engine is a broad-topic query, we can expect to find thousands of relevant pages. The fundamental difficulty here is the *Abundance Problem: The number of pages that could be returned as relevant is too large for a human user to digest* [5]. Often users are not interested in all the relevant pages. Instead they want to see the most *authoritative* or *definitive* pages first. To this end, search engines use ranking algorithms to sort the pages they are going to return as the results of the queries.

Two main classes of algorithms are used for ranking web pages and documents. The first class uses the traditional information retrieval algorithms that are changed to be used in web search engines, while the second class is specifically designed for search engines.

The traditional information retrieval algorithms mostly use Boolean modeling or Vector-Space modeling or an altered version of one of these algorithms for ranking the returned documents. In general, these algorithms do not act successfully in ranking web pages, as they only use the text content of pages for sorting them. What is neglected in these algorithms is that the web is more than a simple set of documents: *web is a set of linked documents.* The ranking methods, which are designed specifically for the web, use this property for ranking the web pages and documents.

2 Web Ranking Algorithms

The algorithms designed for ranking web pages again can be divided to two categories. The algorithms in the first category rank the web pages once, offline and irrespective of the query. Search engines that use these algorithms use specific methods (such as similarity search methods) to find the relative documents to the query and use the ranks computed by the ranking algorithm for sorting these documents. The most famous algorithm in this category is PageRank [4, 3], which is proposed by Page and Brin in Stanford University and is used in Google [2] search engine. The algorithms in the second category rank pages with respect to the query. The most important algorithm in this category is Kleinberg algorithm [5].

2.1 PageRank

PageRank is one the first and most successful algorithms designed for ranking web pages. The main idea of the algorithm is: *A link from page u to page v denotes that the creator of page u has in some measure conferred authority on v.* A page has a high rank if the sum of the ranks of its back links is high. A high rank can be because of high number of back links or highly ranked back links.

The PageRank algorithms is designed as follows:

We assume page A has pages $T_1, \dots T_n$, which point to it. The parameter d is a damping factor, which can be set between 0 and 1. $C(A)$ is defined as the number of links going out of page A. The PageRank of a page A is given as follows:

$$PR(A) = (1 - d) + d(PR(T1)/C(T1) + ... + PR(Tn)/C(Tn))$$

The PageRanks form a probability distribution over web pages, so the sum of all web pages' PageRanks will be one.

PageRank can be thought of as modeling the behavior of a *random surfer.* The random surfer simply keeps clicking on successive links at random. The surfer periodically gets bored and jumps to a random page. The probability that the random surfer visits a page is its PageRank.

2.2 Kleinberg Algorithm

Kleinberg proposed another algorithm for ranking pages. He suggested query specific ranking. Given a query, Kleinberg algorithm finds authorities and hubs with respect to that query. Authorities are good sources of contents, while hubs are good sources of links. Hubs and authorities exhibit a *mutually reinforcing relationship*: *A good hub is a page that points to many good authorities; A good authority is a page that is pointed to by many good hubs.* To find good authorities and hubs, Kleinberg first makes a base set of pages to work with. The base set must be relatively small, rich in relevant pages and it must contain many of the significant authorities. To find such a collection of pages, Kleinberg first collects the highest ranked pages for the query from a text based search engine such as AltaVista [6]. Then he increases the number of strong authorities in this graph by expanding this set along the links that enter and leave it. Kleinberg associates an authority weight A[.] and a hub weight H[.] with each page. The weights are calculated iteratively:

$$A[p] = \sum_{q \to p} H[q]$$

$$H[p] = \sum_{p \to q} A[q]$$

The top c authorities and top c hubs are those pages that have the highest authority and hub weights at the end of the algorithm.

2.2.1 Extensions to Kleinberg Algorithm

A number of drawbacks have been found in Kleinberg algorithm. Bharat and Henzinger[7] point out three reasons that Kleinberg algorithm doesn't work well in all situations: (1) Mutually reinforcing relationship between hosts, (2) automatically generated links and (3) non-relevant nodes. Also a number of algorithms have been proposed to overcome these difficulties. Bharat and Henzinger suggest eliminating non-relevant nodes from the graph and regulating the influence of a node based on its relevance to the query topic.

Chakrabarti and Dom [8] have proposed an extension to Kelinberg algorithm. They weight the links between pages based on the relevance of the text around the link and the query. The major idea is that the text around the links to a page p is descriptive of the contents of p. They assign a weight w(p, q) to each link from page p to page q and update the hub and authority weights using these weights.

In several other, other changes have been made to these algorithms.

3 Proposed Algorithm

The main advantage of PageRank algorithm over Kleinberg algorithm is its running time. In the PageRank algorithm, the ranks of the pages are calculated once, offline. As a result, sorting the pages can be done in a short time. In contrast, the Kleinberg algorithm has to send the query terms to a text-based search engine, such as AltaVista, receive the search results, extend the root set to the base set again using AltaVista and download all the pages in the base set.

PageRank, on the other hand, has its own drawbacks. In the PageRank algorithm, the heavily linked pages often receive high rank, even for queries for which they have no authority. Besides, a page that is considered important in one domain may not be important in other domains, regardless of the words in its text.

In my work, I propose an algorithm that like PageRank can compute the ranks of pages in a short time, but computes the ranks with respect to the query. To this end, I try to change the PageRank algorithm to compute the ranks of pages with respect to the queries.

3.1 Basic Algorithm

Search engines that use PageRank as the algorithm for ranking the web pages and documents, have a graph model of the whole web at hand. The web pages and documents are the nodes of the graph and the links between the pages constitute the directed edges of the graph.

If we assume that we have the graph model of the web at hand, we can use similarity search algorithms to find a number of pages related to the query topic. We use these pages as the root set of pages we are going to apply our ranking algorithm on. If the number of pages is more than a threshold k, we choose the first k pages as the root set. Since we have found these pages using similarity search algorithms, we are sure that the pages are somehow related to the query topic.

In the next step, we try to extend our root set to the base set of pages. We can use the idea of Kleinberg algorithm for our extension. We add the pages that point to the pages of our root set and the pages that are pointed to by the pages of the root set to achieve the base set. In the construction of the base set, we add the restriction that each page S can add at most d pages (which have links to it) to the base set. This restriction is essential, since we are trying to keep our base set small.

We use the induced subgraph of the graph model of the web over the nodes of our base set as the subgraph we are going to compute the ranks on. Since we have assumed that we have the graph model of the web, we can find the subgraph of the base set easily in a short time.

In our basic algorithm, we use the PageRank algorithm to compute the ranks of the the pages of the base set.

3.1.1 Formal Model

In our basic algorithm, the rank of a page p with respect to query Q is defined as follows:

$$R^n(p,Q) = (1-d)\sum_{q\to p}\frac{R^{n-1}(q,Q)}{N_q} + dE(p)$$
$$\pi_{p,Q} = \lim_{n\to\infty} R^n(p,Q)$$

In this equation, $R^n(p,Q)$ is the basic rank of page p with respect to query Q in the n^{th} step, N_q is the number of outgoing links of page q and E is a vector over the web pages corresponding to a source of rank.

The definition of ranks in this way corresponds to a random walk on graphs. The random surfer traverses the graph as follows: In each step, the random surfer selects one of the outgoing links of the current page with probability $(1-d)$ and jumps to a random page (with respect to the source vector E) with probability d. In our basic algorithm, we define E to be uniform over all web pages ($\forall\ 1 \le i \le N, e_i = \frac{1}{N}, N$ is the number of nodes in the base set).

It is fairly easy to show that:

Theorem: *For each query Q and parameter $d > 0$, the basic algorithm finds a unique limiting distribution R^* for the ranks of pages, provided that each page has at least one outgoing edge.*

3.2 Weighting Edges and Nodes of the Neighborhood Graph

For improving the performance of the proposed algorithm, we modify some aspects of the algorithm.

In the first step, we give weight to the links between pages, based on the similarity of the anchor text of the link and the query topic.

3.2.1 Computing the Edge Weights

In general, the links that come out of a specific page are not equally important. In order to enter the importance of links in computing the ranks of pages, we give weight to edges and use these weights when computing the ranks of pages.

For weighting the edges, we use this basic rule: *The text around a link which points to a page p is descriptive of the contents of p.* Note that the sources of these links are not in p, but they are in pages which have links to p.

We give a positive weight $Edge\_wt(p,q)$ to each link between pages p and q. The more the text around the link is related to the query topic, the higher the $Edge\_wt(p,q)$ will be. In order to find the edge weights, we define an anchor window for each link. An anchor window is a window of size $2B$ bytes, B bytes in either side of the start point of the anchor text.

The weights of the edges can be computed using either of these two methods:

1. Suppose $n(t)$ is the number of terms in the anchor window which match with the words of the query. Then we have:
$$Edge\_wt(p \rightarrow q) = 1 + n(t)$$
2. We define the edge weight to be the text similarity of the anchor window and the query:
$$Edge\_wt(p \rightarrow q) = Similarity(Anchor\_Window, Q)$$

On the other hand, if a number of links in one page in the first host all point to pages in a second host, we give fractional edge weights to the outgoing links, so that we will not give undue weight to the opinion of one person of organization. In cases where a number of links from pages in a first host all point to a paper in a second host, we reduce edge weights in the same way.

3.2.2 Computing the Node Weights

The other modification we make in the algorithm is giving weights to pages and documents (nodes of the subgraph) based on their relevance to the query topic. In this way, we can eliminate the nodes that are in the base set, but are not relevant to the query. Besides, we can use these weights when we update the ranks.

For weighting the nodes, we can use one of these two methods:

1. We use the similarity of the text of the page with the extended query[1] as the relevance weight:
$$Node\_wt(p,Q) = Similarity(p,Q)$$
2. We use the anchor text of the links to find the relevance of the node to the query. To this end, we concatenate the anchor texts of all links pointing to page p, and make the anchor document. We use the similarity of this anchor document and the extended query as the relevance weight of the node:
$$Node\_wt(p,Q) = Similarity(Anchor\_doc(p), Q)$$

3.2.3 Pruning nodes from the base set

In some cases, when extending the root set to the base set, unrelated pages are added to the neighborhood graph. We can use the weights of the pages to prune these unrelated nodes. For pruning nodes, we first find a threshold for the weights using one of these three ways:

1. *Median weight:* The threshold is the median of the weights of the nodes.
2. *Root set median weight:* The threshold is the median of the weight of the nodes in the root set.
3. *Fraction of maximum weight:* The threshold is a fixed fraction of the maximum weight (for example $max/10$).

We eliminate those nodes of the base set whose weights are below the defined threshold.

[1] In most cases, the query topic is broader than the query itself. Therefore, we extend our query. In exact words, we concatenate the first 1000 words of each document in the root set to make the extended query Q.

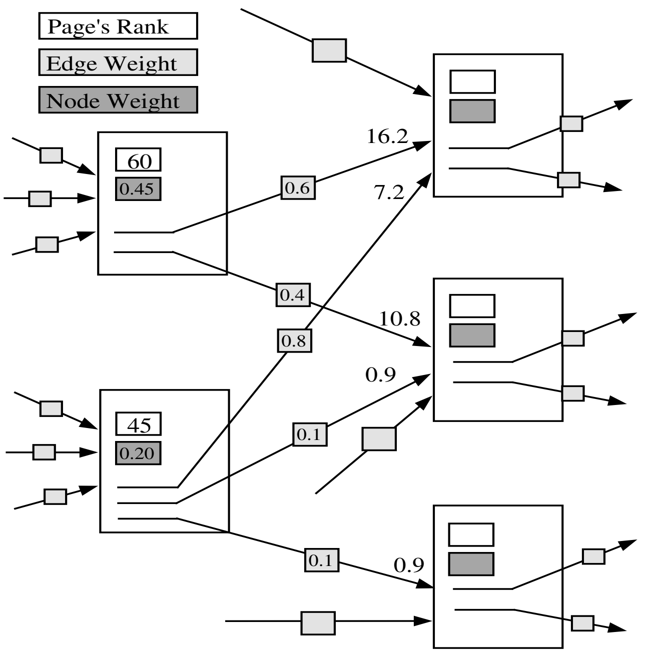

Figure 1: Updating ranks using node and edge weights.

3.3 Updating ranks using edge and node weights

In the weighted subgraph, we compute the ranks of pages using weights of nodes and edges. When updating ranks, we first multiply the weight of the node in its rank and then divide the new rank over web pages. Given the page u and the pages that have links from u $(v_1, \ldots, v_n)$, the fraction of rank of u that is given to v_i is computed as: $Rank(u) \times Node\_wt(u, Q) \times Edge\_wt(u \rightarrow v_i)$. See figure 1 as an example.

3.3.1 Formal model

In our algorithm, we compute the rank of a page p with respect to query Q using this equation:

$$R^n(p,Q) = (1-d) \sum_{q \rightarrow p} R^{n-1}(q,Q) Edge\_wt(q \rightarrow p) Node\_wt(q,Q) + dE(p)$$

Where $R^n(p,Q)$ is the rank of page p with respect to query Q in step n, $Edge\_wt(q \rightarrow p)$ is the relevance of the text of anchor window with query Q, $Node\_wt(q,Q)$ is the relevance weight of node q with respect to query Q and E is a source of rank. For each page p_i, the entry of E corresponding to p_i (e_i) is the similarity of text of p_i and query Q $(Similarity(p_i, Q))$.

The final rank of page p with respect to query Q is defined as:

$$\pi_{p,Q} = \lim_{n \rightarrow \infty} R^n(p,Q)$$

3.3.2 Convergence of Ranks

In order to compute the ranks of pages, we construct a matrix corresponding to the weighted subgraph of our query and multiply this matrix in the rank vector repeatedly until the ranks of pages converge to a limit.

Suppose that query Q and parameter $d > 0$ are given. We construct matrix U as follows: In the first step, if there is no edge from p_i to p_j, we set $u_{ij} = 0$, otherwise we define u_{ij} to be $(1-d) \times Edge\_wt(i \rightarrow j)$. Then for each node j, we add $d \times E(j)$ to each entry u_{ij}, $1 \leq i \leq n$. In the last step, for each i, $1 \leq i \leq n$, we multiply $Node\_wt(i, Q)$ in each entry u_{ij}, $1 \leq j \leq n$.

U is a square matrix whose entries are all positive. We will make use of this property of U to prove the convergence of the ranks in the proposed algorithm.

In order to find the ranks of pages, we first define R^0 as a vector with unique entries. As a result, all pages have equal ranks at the beginning. At each step we find the new ranks by multiplying the transitive of matrix U in the previous vector of ranks. We repeat this task until the ranks converge to a limit:

$$R^n = U^T R^{n-1}$$

$$R^* = \lim_{n \to \infty} R^n$$

Theorem: *With enough number of iterations, the ranks converge to a limit.*

Proof: We use these to theorems to prove the convergence of ranks:

The Theorem of Perron[1]: *If A is a positive matrix, there is a unique characteristic root of A, $\lambda(A)$, which has greatest absolute value. This root is positive and simple, and its associated characteristic vector may be taken to be positive.*

Theorem[1]: *Let c be any non-negative vector. Then*

$$v = \lim_{n \to \infty} A^n(c)/\lambda(A)^n$$

exists and is a characteristic vector of A associated with $\lambda(A)$, unique up to a scalar multiple determined by the choice of c, but otherwise independent of the initial state c.

In our algorithm, U^T is a positive matrix $(\forall i, j, u_{ij} > 0)$. As $n \to \infty$, according to the theorem, the asymptotic behavior of $R(n)$ is given by the relation:

$$R(n) \sim \lambda^n \gamma$$

where λ is the Perron root and γ is the characteristic vector associated with λ. We know that γ is a positive vector which is a positive multiple of one particular normalized characteristic vector δ. The normalization can be taken to be the condition that sum of the components totals n. The constant of proportionality γ and δ is then determined by the values of the initial components of c.

What this means is that independent of the initial vector, as long as U^T is positive, as the number of iterations increases, we approach a steady ranking in which the total ranks of pages grow exponentially, but the proportions of ranks of various papers remain constant. The resulting vector is the ranking vector we were looking for.$\square$

4 Experimental Results

In order to test the proposed algorithm, we implemented a small model of it. We designed a small system for searching the web, with our ranking algorithm as a part of it.

As described earlier, we had assumed that we have the graph model of web at hand. To this end, we downloaded a small subset of the pages of the web and stored the pages in a local storage.

In order to test the proposed algorithm, we need a similarity search algorithm which returns the related pages to the query topic. In our system, we used an implemented similarity search algorithm that returns the result pages. We sort the pages with our ranking algorithm before proposing the results to the user.

In order to store the web pages in a local storage, we used Webzip4.0 software. This software gets the URL of a webpage as input and stores the pages that are up to a user defined level in a local storage. We used http://citeseer.nj.nec.com/brin98anatomy.html as the start URL and saved pages up to five levels.

As our first query, we gave "PageRank" to the system. The first ten returned results were:

Rank	URL	Topic
1	http://citeseer.nj.nec.com/ page98pagerank.html	The PageRank Citation Ranking: Bringing Order to the Web
2	http://citeseer.nj.nec.com/ 345917.html	The PageRank Citation Ranking: Bringing Order to the Web
3	http://citeseer.nj.nec.com/ context/304344/345917.html	Citation: The PageRank Citation Ranking: Bringing Order to the Web
4	http://citeseer.nj.nec.com/ brin98anatomy.html	The Anatomy of a Large-Scale Hypertextual Web Search Engine
5	http://citeseer.nj.nec.com/ 528165.html	PageRank Computation and the Structure of the Web
6	http://citeseer.nj.nec.com/ 527984.html	Efficient Computation of PageRank
7	http://citeseer.nj.nec.com/ 528512.html	Topic-Sensitive PageRank
8	http://citeseer.nj.nec.com/ 356233.html	Data Engineering
9	http://citeseer.nj.nec.com/ pandurangan02using.html	Using PageRank to Characterize Web Structure
10	http://citeseer.nj.nec.com/ 460350.html	The Intelligent Surfer: Probabilistic Combination of Link and Content Information in PageRank

If we study the result pages with care, we will see that the pages are relevant to the query in a good extent. The first result (The PageRank Citation Ranking: Bringing Order to the Web) is the page which describes the basic algorithm of PageRank. The other pages are those which propose extensions to the basic algorithm, present methods for efficient computation of PageRank or use PageRank in different applications of the web.

It can be seen that more important pages and the pages which are referred more are given higher ranks. But there is no accurate metric for evaluation of the algorithm.

In order to ensure that our ranking algorithm ranks pages near reality, we tried to compare our system's output with Google's results. In this comparison, we consider the average difference between the ranks of two algorithms.

If we assume that $R_G(P_i, Q)$ is the priority of P_i in Google's result, $R_P(P_i, Q)$ is the priority of P_i in our algorithm and Pages is the list of returned pages, we define:

$$Difference = 1/n \sum_{i=1}^{n} |R_G(Pages[i]) - R_P(Pages[i])|$$

For our first query ("PageRank"), the difference for the first fifty results was 2.28. This result shows that our algorithm finds the relative arrangement of pages well. Note that we didn't expect the difference to be zero. We compared our algorithm with Google, only to show that the results are not far from reality.

We issued the second query as "Focused Crawling". The first ten results returned by our search system were:

Rank	URL	Title
1	http://citeseer.nj.nec.com/ chakrabarti99focused.html	Focused crawling: a new approach to topic-specific Web resource discovery
2	http://citeseer.nj.nec.com/ 527114.html	Searching the Web
3	http://citeseer.nj.nec.com/ 275140.html	Focused Crawling Using Context Graphs
4	http://citeseer.nj.nec.com/ diligenti00focused.html	Focused Crawling Using Context Graphs
5	http://citeseer.nj.nec.com/ chakrabarti02accelerated.html	Accelerated Focused Crawling through Online Relevance Feedback
6	http://citeseer.nj.nec.com/ chakrabarti99distributed.html	Distributed Hypertext Resource Discovery Through Examples
7	http://citeseer.nj.nec.com/ 271585.html	Topical Locality in the Web
8	http://citeseer.nj.nec.com/ menczer01evaluating.html	Evaluating Topic-Driven Web Crawlers
9	http://citeseer.nj.nec.com/ dlrg02background.html	Background Readings for Collection Synthesis
10	http://citeseer.nj.nec.com/ chakrabarti02structure.html	The Structure of Broad Topics on the Web

For this query, the average difference was 3.1 for our first fifty results.

As noted earlier, the goal of our proposed algorithm was overcoming the drawbacks of PageRank and Kleinberg algorithms. Experiments show that we have achieved our goals.

The main goal of our algorithm was changing the PageRank algorithm, so that it will find the results based on query topic. In PageRank, the algorithm finds the ranks once, offline and uses these ranks in order to sort the query results. The advantage of our algorithm over PageRank is that it can find the ranks based on queries. As a result, the relative arrangement of pages is not fixed as in PageRank algorithm. To experiment this property, we issued the two queries "PageRank, Rank, Ranking" and "HITS, Kleinberg, Rank, Ranking" to the search system. the two pages http://www.citeseer.nj.nec.com/page98pagerank.html and http://citeseer.nj.nec.com/kleinberg97authoritative.html were in the results of both queries. In the first query, page98pagerank.html had higher priority and in the second query, kleinberg97authoritative.html was ranked higher.

5 Conclusion

In this paper, we introduced a new ranking algorithm, which exploits hyperlink as well as document contents for ranking resources, with the intent of overcoming the drawbacks of current ranking algorithms. We showed that our ranking algorithm converges. We also implemented a small model of our algorithm. The results show that our method fulfills our goals.

References

[1] Richard Bellman, *Introduction to Matrix Analysis*, siam, 1997.

[2] http://www.google.com

[3] Sergey Brin, Larry Page, *The Anatomy of a Large-Scale Hypertextual Web Search Engine*, In Proceedings of the Seventh International World Wide Web Conference, 1998.

[4] Sergey Brin, Larry Page, *The PageRank Citation Ranking : Bringing Order to the Web*, Technical Report, Stanford University, 1998.

[5] Jon M. Kleinberg, *Authoritative Sources in a Hyperlinked Environment* In Proceedings of the ACM-SIAM Symposium on Discrete Algorithms, 1998.

[6] http://www.altavista.com

[7] Krishna Bharat, Monika R. Henzinger, *Improved Algorithms for Topic Distillation in a Hyperlinked Environment*, In Proceedings of the 21st International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 104-111, Aug. 1998.

[8] S. Chakrabarti, B. Dom, P. Raghavan, S. Rajagonpalan, D. Gibson, J. Kleinberg, *Automatic Resource Compilation by Analyzing Hyperlink Structure and Associated Text*, In Proc. 7th International World Wide Web Conference, 1998.