International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:3, No:4, 2009

A Comparative Study of Page Ranking Algorithms for Information Retrieval

Ashutosh Kumar Singh, Ravi Kumar P

***Abstract*—**This paper gives an introduction to Web mining, then describes Web Structure mining in detail, and explores the data structure used by the Web. This paper also explores different Page Rank algorithms and compare those algorithms used for Information Retrieval. In Web Mining, the basics of Web mining and the Web mining categories are explained. Different Page Rank based algorithms like PageRank (PR), WPR (Weighted PageRank), HITS (Hyperlink-Induced Topic Search), DistanceRank and DirichletRank algorithms are discussed and compared. PageRanks are calculated for PageRank and Weighted PageRank algorithms for a given hyperlink structure. Simulation Program is developed for PageRank algorithm because PageRank is the only ranking algorithm implemented in the search engine (Google). The outputs are shown in a table and chart format.

***Keywords*—**Web Mining, Web Structure, Web Graph, Link Analysis, PageRank, Weighted PageRank, HITS, DistanceRank, DirichletRank,

I. INTRODUCTION

THE World Wide Web (WWW) is growing tremendously on all aspects and is a massive, explosive, diverse, dynamic and mostly unstructured data repository. As on today WWW is the largest information repository for knowledge reference. There are a lot of challenges [1] in the Web: Web is huge, Web pages are semi-structured, Web information tends to be diversity in meaning, degree of quality of the information extracted and the conclusion of the knowledge from the extracted information. A Google report [5] on 25th July 2008 says that there are 1 trillion (1,000,000,000,000) unique URLs (Universal Resource Locator) on the Web. The actual number could be more than that and Google could not index all the pages. When Google first created the index in 1998 there were 26 million pages and in 2000 Google index reached 1 billion pages. In the last 9 years, Web has grown tremendously and the usage of the web is unimaginable. So it is important to understand and analyze the underlying data structure of the Web for effective Information Retrieval. Web mining techniques along with other areas like Database (DB), Information Retrieval (IR), Natural Language Processing (NLP), Machine Learning etc. can be used to solve the above challenges.

With the rapid growth of WWW and the user's demand on knowledge, it is becoming more difficult to manage the information on WWW and satisfy the user needs. Therefore, the users are looking for better information retrieval techniques and tools to locate, extract, filter and find the necessary information. Most of the users use information retrieval tools like search engines to find information from the WWW. There are tens and hundreds of search engines available but some are popular like Google, Yahoo, Bing etc., because of their crawling and ranking methodologies. The search engines download, index and store hundreds of millions of web pages. They answer tens of millions of queries every day. So Web mining and ranking mechanism becomes very important for effective information retrieval. The sample architecture [2] of a search engine is shown in Fig. 1.

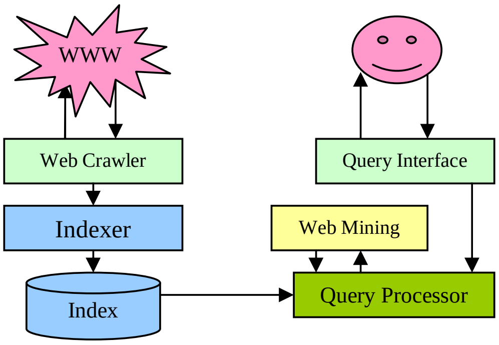


Fig. 1 Sample Architecture of a Search Engine

There are 3 important components in a search engine. They are Crawler, Indexer and Ranking mechanism. The crawler is also called as a robot or spider that traverses the web and downloads the web pages. The downloaded pages are sent to an indexing module that parses the web pages and builds the index based on the keywords in those pages. An alphabetical index is generally maintained using the keywords. When a user types a query using keywords on the interface of a search engine, the query processor component match the query keywords with the index and returns the URLs of the pages to the user. But before presenting the pages to the user, a ranking mechanism is done by the search engines to present the most relevant pages at the top and less relevant ones at the bottom. It makes the search results navigation easier for the user. The ranking mechanism is explained in detail later in this paper.

This paper is organized as follows. Section II provides the basic Web mining concepts and the three areas of Web mining. In this section Web Structure mining is described in detail because most of the Page Rank algorithms are based on

Ashutosh Kumar Singh is with the Department of Electrical and Computer Engineering, Curtin University of Technology, Miri, Sarawak, Malaysia (e-mail: ashutosh.s@curtin.edu.my).

Ravi Kumar P is with the Department of Computer & I.T., Jefri Bolkiah College of Engineering, Brunei, doing his PhD at Curtin University of Technology, Miri, Malaysia (e-mail: ravi2266@gmail.com).

the Web Structure Mining. Section III describes Data Structure used for Web in particular the *Web Graph*. Section IV explores important algorithms based on Page Rank and compares those algorithms. Section V shows the simulation results and Section VI concludes this paper.

II. Web Mining

A. Overview

Web mining is the use of data mining techniques to automatically discover and extract information from the World Wide Web (WWW). According to Kosala et al [3], Web mining consists of the following tasks:

- *Resource finding*: the task of retrieving intended Web documents.
- *Information selection and pre-processing*: automatically selecting and pre-processing specific information from retrieved Web resources.
- *Generalization*: automatically discovers general patterns at individual Web sites as well as across multiple sites.
- *Analysis*: validation and/or interpretation of the mined patterns.

Resource finding is the process of retrieving the data that is either online or offline from the electronic newsgroups, newsletters, newswire, Libraries, HTML documents that are available as text sources on the Web. Information selection and pre-processing is selecting the HTML documents and transform the HTML documents by removing HTML tags, stop words, stemming etc. Generalization is the process of discovering general patterns at individual web sites as well as across multiple sites. Analysis referred to the validation and/or interpretation of the mined patterns. Human plays an important role in the information or knowledge discovery process on the Web since Web is an interactive medium. This is especially important for validation and/or interpretation.

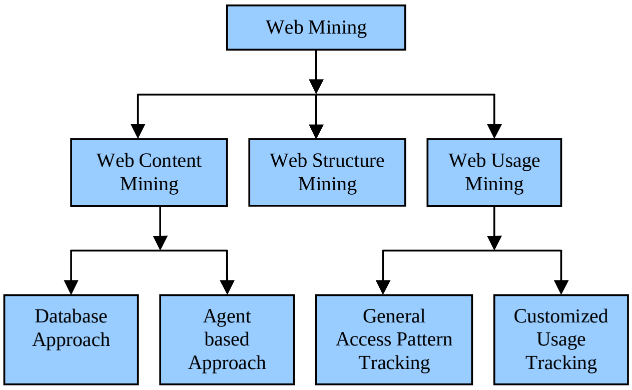


Fig. 2 Web Classification

There are three areas of Web mining according to the usage of the Web data used as input in the data mining process, namely, Web Content Mining (WCM), Web Usage Mining (WUM) and Web Structure Mining (WSM). Web content mining is concerned with the retrieval of information from WWW into more structured form and indexing the information to retrieve it quickly. Web usage mining is the process of identifying the browsing patterns by analyzing the user's navigational behavior. Web structure mining is to discover the model underlying the link structures of the Web pages, catalog them and generate information such as the similarity and relationship between them, taking advantage of their hyperlink topology. Web classification [4] is shown in Fig 2. Even though there are three areas of Web mining, the differences between them are narrowing because they are all interconnected. Web Content mining and Web Structure mining are related. They are basically used to extract the knowledge from the World Wide Web. Web content is concerned with the retrieval of information from WWW into more structured form. Web structure mining helps to retrieve more relevant information by analyzing the link structure. Most of the researchers now focus on the combination of the three areas of Web mining to produce a better research.

Definitions

Web Content Mining (WCM)

Web Content Mining is the process of extracting useful information from the contents of web documents. The web documents may consists of text, images, audio, video or structured records like tables and lists. Mining can be applied on the web documents as well the results pages produced from a search engine. There are two types of approach in content mining called agent based approach and database based approach. The agent based approach concentrate on searching relevant information using the characteristics of a particular domain to interpret and organize the colleted information. The database approach is used for retrieving the semi-structure data from the web.

Web Usage Mining (WUM)

Web Usage Mining is the process of extracting useful information from the secondary data derived from the interactions of the user while surfing on the Web. It extracts data stored in server access logs, referrer logs, agent logs, client-side cookies, user profile and meta data.

Web Structure Mining (WSM)

The goal of the Web Structure Mining is to generate the structural summary about the Web site and Web page. It tries to discover the link structure of the hyperlinks at the inter-document level. Based on the topology of the hyperlinks, Web Structure mining will categorize the Web pages and generate the information like similarity and relationship between different Web sites. This type of mining can be performed at the document level (intra-page) or at the hyperlink level (inter-page). It is important to understand the Web data structure for Information Retrieval.

III. Data Structure for Web

The traditional information retrieval system basically focuses on information provided by the text of Web documents. Web mining technique provides additional information through hyperlinks where different documents are connected. The Web may be viewed as a directed labeled graph whose nodes are the documents or pages and the edges are the hyperlinks between them. This directed graph structure in the Web is called as *Web Graph.*

A graph *G* consists of two sets *V* and *E*, Horowitz et al [6]. The set *V* is a finite, nonempty set of *vertices*. The set *E* is a set of pairs of vertices; these pairs are called *edges*. The notation *V*(*G*) and *E*(*G*) represent the sets of vertices and edges, respectively of graph *G*. It can also be expressed *G* = (*V*, *E*) to represent a graph. In an *undirected graph* the pair of vertices representing any edge is unordered. Thus the pairs (*u*, *v*) and (*v*, *u*) represent the same edge. In a *directed graph* each edge is represented by a directed pair (*u*, *v*); *u* is the tail and *v* is the head of the edge. Therefore, (*v*, *u*) and (*u*, *v*) represent two different edges. The graph in Fig. 3 is a directed graph with 3 Vertices and 6 edges.

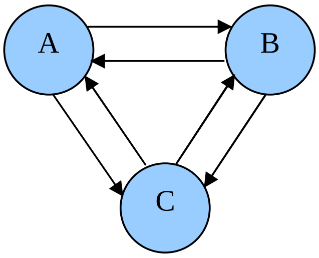


Fig. 3 A Directed Graph, G

The vertices *V* of *G*, *V*(*G*) = {*A*, *B*, *C*}. The Edges *E* of *G*, *E*(*G*) ={(*A*, *B*), (*B*, *A*), (*B*, *C*), (*C*, *B*), (*A*, *C*), (*C*, *A*)}. In a directed graph with *n* vertices, the maximum number of edges is *n*(*n*-1). With 3 vertices, the maximum number of edges can be 3(3-1) = 6. A directed graph is said to be *strongly connected* if for every pair of distinct vertices *u* and *v* in *V*(*G*), there is a directed path from *u* to *v* and also from *v* to *u*. The above graph in Fig. 3 is strongly connected, because all the vertices are connected in both directions. According to Broader et al. [7], a Web can be imagined as a large graph containing several hundred million or billion of nodes or vertices, and a few billion arcs or edges. The degree of a vertex is the number of edges incident to that vertex. If *G* is a directed graph, we define the *in-degree* of a vertex *v* to be the number of edges for which *v* is the head. The *out-degree* is defined to be the number of edges for which *v* is the tail. In Fig. 3, the graph *G*, vertex *B* has in-degree 2, out-degree 2 and degree 4. If d_i is the degree of vertex *i* in a graph *G* with *n* vertices and *e* edges, then the number of edges is as shown in (1).

$$e = \sum_{i=1}^{n} (d_i)/2 \tag{1}$$

Several researches have done to analyze the properties of the *graph* [8, 9]. Broader et al. showed the structure of the Web graph looks like a giant bow tie as shown in Fig. 4. This, Web macroscopic structure has four pieces. The first piece is a central core, all of whose pages can reach one another along directed links -- this "giant strongly connected component" (*SCC*) is at the heart of the web. The second and third pieces are called *IN* and *OUT*. *IN* consists of pages that can reach the *SCC*, but cannot be reached from it - possibly new sites that people have not yet discovered and linked to. *OUT* consists of pages that are accessible from the *SCC*, but do not link back to it, such as corporate websites that contain only internal links. Finally, the *TENDRILS* contain pages that cannot reach the *SCC*, and cannot be reached from the *SCC*. The macroscopic structure of the Web [7] is shown in Fig. 4. According to Broader et al., the size of the *SCC* is relatively small comparing with *IN*, *OUT* and *Tendrils*. Almost all the sets have roughly the same size. So it is evident that the Web is growing rapidly and it is a huge structure. The following section explains important page ranking algorithms and compares those algorithms used for information retrieval.

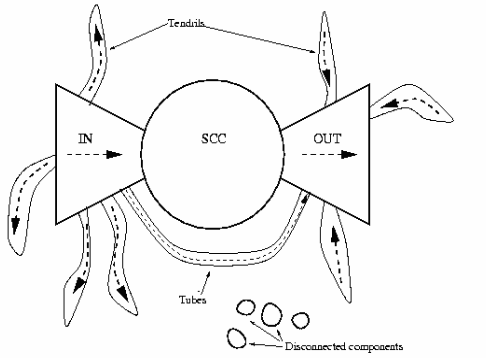


Fig. 4 Macroscopic Structure of Web [7]

IV. Page Rank Algorithms

With the increasing number of Web pages and users on the Web, the number of queries submitted to the search engines are also increasing rapidly. Therefore, the search engines needs to be more efficient in its process. Web mining techniques are employed by the search engines to extract relevant documents from the web database and provide the necessary information to the users. The search engines become very successful and popular if they use efficient Ranking mechanism. Google search engine is very successful because of its *PageRank* algorithm. Page ranking algorithms are used by the search engines to present the search results by considering the relevance, importance and content score and web mining techniques to order them according to the user interest. Some ranking algorithms depend only on the link structure of the documents i.e. their popularity scores (web structure mining), whereas others look for the actual content in the documents (web content mining), while some use a combination of both i.e. they use content of the document as well as the link structure to assign a rank value for a given document. If the search results are not displayed according to the user interest then the search engine will loose its popularity. So the ranking algorithms become very important.

Some of the popular page ranking algorithms are discussed in the following section.

A. Citation Analysis

Link analysis is similar to social networks and citation analysis. The citation analysis was developed in information science as a tool to identify core sets of articles, authors, or journals of a particular field of study. "Impact factor" [10] developed by Eugene Garfield is used to measure the importance of a publication. This metric takes into account the number of citations received by a publication. The impact factor is proportional to the total number of citations a publications has. This treats all the references equally. Some important references which are referred many times should be given an additional weight. Pinski et al [11] proposed a model to overcome this problem called "influence weights" where the weight of each publication is equal to the sum of its citations, scaled by the importance of these citations.

The same principle is applied to the Web for ranking the web pages where the notion of citations corresponds to the links pointing to a Web page. This simplest ranking of a Web page could be done by summing up the number of links pointing to it. This favors only the most popular Web sites, such as universally known portals, news pages, news broadcasters etc. In the Web, the quality of the page and the content's diversity should also be considered. In the scientific literature, co-citations are between the same networks of knowledge. On the other hand, Web contains lot of information, serving for different purposes.

These "hyperlinked communities that appear to span a wide range of interests and disciplines", Gibson et al [12] are called as "Web communities" and the process of identifying them is called as "trawling", R. Kumar et al [13].There are a number of algorithms proposed based on the Link Analysis. Using citation analysis, Co-citation algorithm [14] and Extended Co-citation algorithm [15] are proposed. These algorithms are simple and deeper relationships among the pages can not be discovered.

Five Page Rank based algorithms *PageRank [PR]* [16], *Weighted PageRank* (*WPR*) [17], Hypertext Induced Topic Search *HITS* [18], *DistanceRank* [23] and *DirichletRank* [27] algorithms are discussed below in detail and compared.

B. PageRank

Brin and Page developed *PageRank* algorithm during their Ph D at Stanford University based on the citation analysis. *PageRank* algorithm is used by the famous search engine, Google. They applied the citation analysis in Web search by treating the incoming links as citations to the Web pages. However, by simply applying the citation analysis techniques to the diverse set of Web documents did not result in efficient outcomes. Therefore, *PageRank* provides a more advanced way to compute the importance or relevance of a Web page than simply counting the number of pages that are linking to it (called as "backlinks"). If a backlink comes from an "important" page, then that backlink is given a higher weighting than those backlinks comes from non-important pages. In a simple way, link from one page to another page may be considered as a vote. However, not only the number of votes a page receives is considered important, but the "importance" or the "relevance" of the ones that cast these votes as well.

Assume any arbitrary page *A* has pages T_1 to T_n pointing to it (incoming link). *PageRank* can be calculated by the following (2).

$$PR(A) = (1-d) + d(PR(T_1)/C(T_1) + ... + PR(T_n / C(T_n)) \qquad (2)$$

The parameter *d* is a damping factor, usually sets it to 0.85 (to stop the other pages having too much influence, this total vote is "damped down" by multiplying it by 0.85). *C(A)* is defined as the number of links going out of page *A*. The *PageRanks* form a probability distribution over the Web pages, so the sum of all Web pages' *PageRank* will be one. *PageRank* can be calculated using a simple iterative algorithm, and corresponds to the principal eigenvector of the normalized link matrix of the Web.

Let us take an example of hyperlink structure of four pages *A*, *B*, *C* and *D* as shown in Fig. 5. The *PageRank* for pages *A*, *B*, *C* and *D* can be calculated by using (2).

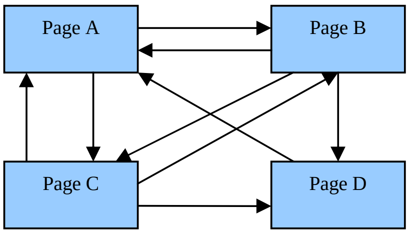


Fig. 5 Hyperlink Structure for 4 pages

Let us assume the initial *PageRank* as 1 and do the calculation. The damping factor *d* is set to 0.85.

$$PR(A) = (1\text{-}d) + d\ (PR(B)/C(B)+PR(C)/C(C)+PR(D)/C(D))$$
$$= (1\text{-}0.85) + 0.85(1/3+1/3+1/1) = 1.566667 \qquad (2a)$$
$$PR(B) = (1\text{-}d) + d((PR(A)/C(A) + (PR(C)/C(C)) = 1.099167 \quad (2b)$$
$$PR(C) = (1\text{-}d) + d((PR(A)/C(A) + (PR(B)/C(B)) = 1.127264 \quad (2c)$$
$$PR(D)= (1\text{-}d) + d((PR(B)/C(B) + (PR(C)/C(C)) = 0.780822 \quad (2d)$$

Do the second iteration by taking the above *PageRank* values from (2a), (2b), (2c) and 2(d).

$$PR(A) = 0.15 + 0.85((1.099167/3)+(1.127264/3)+(0.780822/1)$$
$$= 1.444521 \qquad (2e)$$
$$PR(B) = 0.15 + 0.85((1.444521/2)+(1.127264/3)) = 1.083313 \quad (2f)$$
$$PR(C) = 0.15 + 0.85((1.444521/2)+(1.083313/3)) = 1.07086 \quad (2g)$$
$$PR(D) = 0.15 + 0.85((1.083313/3)+(1.07086/3)) = 0.760349 \quad (2h)$$

During the 34th iteration, the PageRank gets converged as shown in Table I. The table with the graph is shown in the Simulation Results section.

TABLE I
ITERATIVE CALCULATION FOR PAGERANK

Iteration	A	B	C	D
1	1	1	1	1
2	1.566667	1.099167	1.127264	0.780822
3	1.444521	1.083313	1.07086	0.760349
4	1.406645	1.051235	1.045674	0.744124
..	..	..	..	..
..	..	..	..	..
33	1.31351	0.988244	0.988244	0.710005
34	1.313509	0.988244	0.988244	0.710005
35	1.313509	0.988244	0.988244	0.710005

For a smaller set of pages, it is easy to calculate and find out the *PageRank* values but for a Web having billions of pages, it is not easy to do the calculation like above. In the above Table I, you can notice that *PageRank* of *A* is higher than *PageRank* of *B*, *C* and *D*. It is because Page *A* has 3 incoming links, Page *B*, *C* and *D* have 2 incoming links as shown in Fig. 5. Page B has 2 incoming links and 3 outgoing link. Page C has 2 incoming links and 3 outgoing links. Page *D* has 1 incoming link and 2 outgoing links. From the Table I, after the iteration 34, the *PageRank* for the pages gets normalized. Previous experiments [19, 20] showed that the *PageRank* gets converged to a reasonable tolerance. The convergence of *PageRank* calculation is shown as a graph in Fig.11 in the Simulation Result section.

PageRank is displayed on the toolbar of the browser if the Google Toolbar [32] is installed. The Toolbar *PageRank* goes from 0 – 10, like a logarithmic scale with 0 is the low page rank and 10 is the highest page rank. The *PageRank* of all the pages on the Web changes every month when Google does its re-indexing. *PageRank* says nothing about the content or size of a page, the language it's written in, or the text used in the anchor of a link.

C. Weighted PageRank Algorithm

Wenpu Xing and Ali Ghorbani [17] proposed a *Weighted PageRank* (*WPR*) algorithm which is an extension of the *PageRank* algorithm. This algorithm assigns a larger rank values to the more important pages rather than dividing the rank value of a page evenly among its outgoing linked pages. Each outgoing link gets a value proportional to its importance. The importance is assigned in terms of weight values to the incoming and outgoing links and are denoted as $W^{in}(m, n)$ and $W^{out}(m, n)$ respectively. $W^{in}(m, n)$ as shown in (3) is the weight of *link*(*m, n*) calculated based on the number of incoming links of page *n* and the number of incoming links of all reference pages of page *m*.

$$W^{in}_{(m,n)} = \frac{I_n}{\sum_{p \in R(m)} I_p} \tag{3}$$

$$W^{out}_{(m,n)} = \frac{O_n}{\sum_{p \in R(m)} O_p} \tag{4}$$

Where I_n and I_p are the number of incoming links of page *n* and page *p* respectively. *R*(*m*) denotes the reference page list of page *m*. $W^{out}(m, n)$ is as shown in (4) is the weight of *link*(*m, n*) calculated based on the number of outgoing links of page *n* and the number of outgoing links of all reference pages of *m*. Where O_n and O_p are the number of outgoing links of page *n* and *p* respectively. The formula as proposed by Wenpu et al for the *WPR* is as shown in (5) which is a modification of the *PageRank* formula.

$$WPR(n) = (1-d) + d \sum_{m \in B(n)} WPR(m) W^{in}_{(m,n)} W^{out}_{(m,n)} \tag{5}$$

Use the same hyperlink structure as shown in Fig. 5 and do the *WPR* Calculation. The *WPR* equation for Page *A*, *B*, *C* and *D* are as follows.

$$WPR(A) = (1-d) + d(WPR(B).W^{in}_{(B,A)} \cdot W^{out}_{(B,A)} + WPR(C).W^{in}_{(C,A)} \cdot W^{out}_{(C,A)} \tag{5a}$$
$$+ WPR(D).W^{in}_{(D,A)} \cdot W^{out}_{(D,A)})$$

$$\begin{aligned} WPR(B) = (1-d) + d(WPR(A).W^{in}_{(A,B)} \cdot W^{out}_{(A,B)} \\ + WPR(C).W^{in}_{(C,B)} \cdot W^{out}_{(C,B)}) \end{aligned} \tag{5b}$$

$$\begin{aligned} WPR(C) = (1-d) + d(WPR(A).W^{in}_{(A,C)} \cdot W^{out}_{(A,C)} \\ + WPR(B).W^{in}_{(B,C)} \cdot W^{out}_{(B,C)}) \end{aligned} \tag{5c}$$

$$\begin{aligned} WPR(D) = (1-d) + d(WPR(B).W^{in}_{(B,D)} \cdot W^{out}_{(B,D)} \\ + WPR(C).W^{in}_{(C,D)} \cdot W^{out}_{(C,D)}) \end{aligned} \tag{5d}$$

The incoming link and outgoing link weights are calculated as follows:

$$W^{in}_{(B,A)} = I_A / (I_A + I_C) = 3/(3+2) = 3/5 \tag{5e}$$

$$W^{out}_{(B,A)} = O_A / (O_A + O_C + O_D) = 2/(2+3+1) = 2/6 = 1/3 \tag{5f}$$

$$W^{in}_{(C,A)} = I_A / (I_A + I_B) = 3/(3+2) = 3/5 \tag{5g}$$

$$W^{out}_{(C,A)} = O_A / (O_A + O_B + O_D) = 2/(2+3+1) = 2/6 = 1/3 \tag{5h}$$

$$W^{in}_{(D,A)} = I_A / (I_B + I_C) = 3/(2+2) = 3/4 \tag{5i}$$

$$W^{out}_{(D,A)} = O_A / O_A = 2/2 = 1 \tag{5j}$$

By substituting the values of (5e), (5f), 5(g), 5(h), 5(i) and 5(j) to (5a), you will get the *WPR* of Page *A* by taking a value of 0.85 for *d* and the initial value of *WPR*(*B*), *WPR*(*C*) and *WPR*(*D*) = 1.

$$WPR(A) = (1-0.85) + 0.85(1*3/5*1/3 + 1*3/5*1/3 + 1*3/4*1) = 1.127 \tag{5k}$$

$$W^{in}_{(A,B)} = I_B / (I_B + I_C + I_D) = 2/(2+2+2) = 2/6 = 1/3 \tag{5l}$$

$$W^{out}_{(A,B)} = O_B / (O_B + O_C) = 3/(3+3) = 3/6 = 1/2 \tag{5m}$$

$$W^{in}_{(C,B)} = I_B / (I_A + I_B) = 2/(3+2) = 2/5 \tag{5n}$$

$$W^{out}_{(C,B)} = O_B/(O_A + O_B + O_D) = 3/(2+3+1) = 3/6 = 1/2 \quad (5o)$$

By substituting the values of (5k), (5l), (5m), (5n) and (5o) to (5b), you will get the *WPR* of Page *B* by taking *d* as 0.85 and the initial value of *WPR*(*C*) = 1

$$\begin{aligned} WPR\ (B) &= (1 - 0.85 + 0.85((1.127\ *1/3*1/2+1*2/5*1/2) \\ &= 0.499 \end{aligned} \quad (5p)$$

$$W^{in}_{(A,c)} = I_C/(I_B + I_C + I_D) = 2/(2+2+2) = 2/6 = 1/3 \quad (5q)$$

$$W^{out}_{(A,C)} = O_C/(O_B + O_C) = 3/(3+3) = 3/6 = 1/2 \quad (5r)$$

$$W^{in}_{(B,C)} = I_C/(I_A + I_B) = 2(3+2) = 2/5 \quad (5s)$$

$$W^{out}_{(B,C)} = O_C/(O_A + O_C + O_D) = 3/(2+3+1) == 3/6 = 1/2 \quad (5t)$$

By substituting the values of (5k), (5p), (5q), (5r), (5s) and (5t) to (5c), you will get the *WPR* of Page *C* by taking *d* as 0.85.

$$WPR(C) = (1-0.85) + 0.85((1.127*1/3*1/2) + (0.499*2/5*1/2)) = 0.392 \quad (5u)$$

$$W^{in}_{(B,D)} = I_D/(I_B + I_C) = 2(2+2) = 2/4 = 1/2 \quad (5v)$$

$$W^{out}_{(B,D)} = O_D/O_A = 2/2 = 1 \quad (5w)$$

$$W^{in}_{(C,D)} = I_D/(I_A + I_B) = 2(2+3) = 2/5 \quad (5x)$$

$$W^{out}_{(C,D)} = O_D/(O_A + O_B + O_D) = 2/2+3+1 = 2/6 = 1/3 \quad (5y)$$

By substituting the values of (5p), (5u), (5v), (5w), (5x) and (5y) to (5d), you will get the *WPR* of Page *D* by taking *d* as 0.85.

$$WPR(D) = (1-0.85) + 0.85((0.499*1/2*1) + (0.392*2/5*1/3)) = 0.406 \quad (5z)$$

The values of *WPR*(*A*), *WPR*(*B*), *WPR*(*C*) and *WPR*(*D*) are shown in (5k), (5p), (5u) and (5z) respectively. In this, *WPR*(*A*)>*WPR*(*B*)>*WPR*(*D*)>*WPR*(*C*). This results shows that the page rank order is different from *PageRank*.

To differentiate the *WPR* from the *PageRank*, Wenpu et al, categorized the resultant pages of a query into four categories based on their relevancy to the given query: They are:

1. *Very Relevant Pages* (*VR)*: The pages containing very important information related to a given query.
2. *Relevant Pages* (*R*): Pages are relevant but not having important information about a given query.
3. *Weak Relevant Pages* (*WR*): Pages may have the query keywords but they do not have the relevant information.
4. *Irrelevant Pages* (*IR*): Pages not having any relevant information and query keywords.

The *PageRank* and *WPR* algorithms both provide ranked pages in the sorting order to users based on the given query. So, in the resultant list, the number of relevant pages and their order are very important for users. Wenpu et al proposed a *Relevance Rule* to calculate the relevancy value of each page in the list of pages. That makes *WPR* is different from *PageRank.*

Relevancy Rule: The Relevancy Rule is as shown in equation (6). The Relevancy of a page to a given query depends on its category and its position in the page-list. The larger the relevancy value, the better is the result.

$$k = \sum_{i \in R(p)} (n-i) * W_i \quad (6)$$

Where *i* denotes the i^{th} page in the result page-list *R(p)*, *n* represents the first *n* pages chosen from the list *R(p)*, and W_i is the weight of i^{th} page as given below:

$$W_i = (v1, v2, v3, v4)$$

Where *v*1, *v*2, *v*3 and *v*4 are the values assigned to a page if the page is *VR*, *R*, *WR* and *IR* respectively. The values are always *v*1>*v*2>*v*3>*v*4. Experimental studies by Wenpu et al. showed that *WPR* produces larger relevancy values than the *PageRank.*

C. The HITS Algorithm - Hubs & Authorities

Kleinberg [18] identifies two different forms of Web pages called *hubs* and *authorities*. Authorities are pages having important contents. Hubs are pages that act as resource lists, guiding users to authorities. Thus, a good hub page for a subject points to many authoritative pages on that content, and a good authority page is pointed by many good hub pages on the same subject. Hubs and Authorities are shown in Fig. 6. Kleinberg says that a page may be a good hub and a good authority at the same time. This circular relationship leads to the definition of an iterative algorithm called HITS (Hyperlink Induced Topic Search).

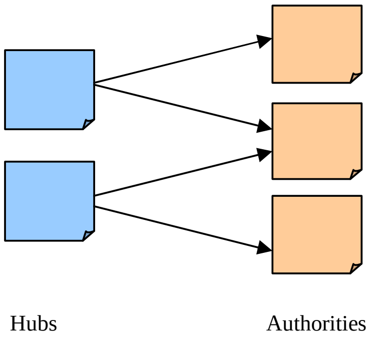


Fig. 6 Hubs and Authorities

The HITS algorithm treats WWW as a directed graph *G*(*V*,*E*), where *V* is a set of Vertices representing pages and *E* is a set of edges that correspond to links.

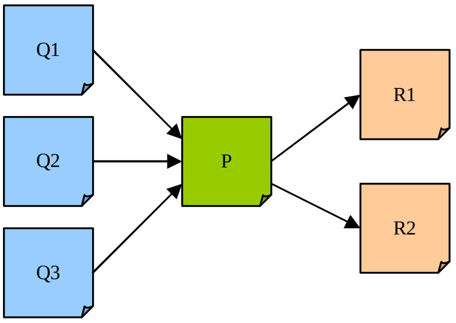
HITS Working Method

There are two major steps in the HITS algorithm. The first step is the *Sampling Step* and the second step is the *Iterative step*. In the *Sampling step*, a set of relevant pages for the given query are collected i.e. a sub-graph *S* of *G* is retrieved which is high in authority pages. This algorithm starts with a root set *R*, a set of *S* is obtained, keeping in mind that *S* is relatively small, rich in relevant pages about the query and contains most of the good authorities. The second step, *Iterative step*, finds hubs and authorities using the output of the sampling step using (7) and (8).

$$H_p = \sum_{q \in I(p)} A_q \qquad (7)$$

$$A_p = \sum_{q \in B(p)} H_q \qquad (8)$$

Where H_p is the hub weight, A_p is the Authority weight, $I(p)$ and $B(p)$ denotes the set of reference and referrer pages of page *p*. The page's authority weight is proportional to the sum of the hub weights of pages that it links to it, Kleinberg [21]. Similarly, a page's hub weight is proportional to the sum of the authority weights of pages that it links to. Fig. 7 shows an example of the calculation of authority and hub scores.



$A_P = H_{Q1} + H_{Q2} + H_{Q3}$ $\qquad$ $H_P = A_{R1} + A_{R2}$

Fig. 7 Calculation of hubs and Authorities

Constraints of HITS

The following are the constraints of HITS algorithm [22]:

- *Hubs and authorities*: It is not easy to distinguish between hubs and authorities because many sites are hubs as well as authorities.
- *Topic drift*: Sometime HITS may not produce the most relevant documents to the user queries because of equivalent weights.
- *Automatically generated links:* HITS gives equal importance for automatically generated links which may not produce relevant topics for the user query.
- *Efficiency*: HITS algorithm is not efficient in real time.

HITS was used in a prototype search engine called Clever [22] for an IBM research project. Because of the above constraints HITS could not be implemented in a real time search engine.

D. DistanceRank Algorithm

DistanceRank algorithm is proposed by Ali Mohammad Zareh Bidoki and Nasser Yazdani [23]. They proposed a novel recursive method based on reinforcement learning [24] which considers distance between pages as punishment, called "*DistanceRank*" to compute ranks of web pages. The number of 'average clicks' between two pages is defined as distance. The main objective of this algorithm is to minimize distance or punishment so that a page with smaller distance to have a higher rank.

Most of the current ranking algorithms have the "rich-get-richer" problem [25] i.e. the popular high rank web pages become more and more popular and the young high quality pages are not picked by the ranking algorithms. There are many solutions [23, 25] suggested for this "rich-get-richer" problem. The *DistanceRank* algorithm algorithms is less sensitive to the "rich-get-richer" problem and finds important pages faster than others. This algorithm is based on the reinforcement learning such that the distance between pages is treated as punishment factor. Normally related pages are linked to each other so the distance based solution can find pages with high qualities more quickly.

In *PageRank* algorithm, the rank of each page is defined as the weighted sum of ranks of all pages having back links or incoming links to the page. Then, a page has a high rank if it has more back links to this page have higher ranks. These two properties are true for *DistanceRank* also. A page having many incoming links should have low distance and if pages pointing to this page have low distance then this page should have a low distance. The above point is clarified using the following definition.

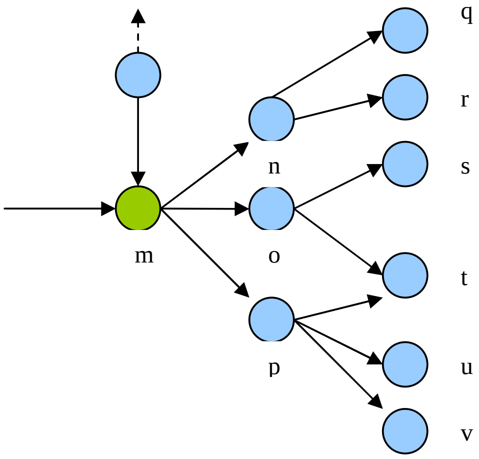


Fig. 8 A Sample Graph

Definition 1. If page *a* points to page *b* then the weight of link between *a* and *b* is equal to $\text{Log}_{10}\text{O}(a)$ where O(*a*) shows *a*'s out degree or outgoing links.

Definition 2. The distance between two pages *a* and *b* is the weight of the shortest path (the path with the minimum value) from *a* to *b*. This is called *logarithmic distance* and is denoted as d_{ab}.

For example, in Fig. 8, the weight of out-links or out going links in pages *m, n, o* and *p* is equal to log(3), log(2), log(2) and log(3) respectively and the distance between *m* and *t* is equal to log(3) + log(2) if the path *m*—*o*—*t* was the shortest path between *m* and *t*. The distance between *m* and *v* is log(3) + log(3) as shown in Fig. 8 even though both *t* and *v* are in the same link level from *m* (two clicks) but *t* is closer to *m*.

*Definition 3***.** If d_{ab} shows the distance between two pages *a* and *b* as Definition 2, then d_b denotes the average distance of page *b* and is defined as the following where *V* shows number of web pages:

$$d_b = \frac{\sum_{a=1}^{v} d_{ab}}{V} \tag{21}$$

In this definition, the authors used an average click instead of the classical distance definition. The weight of each link is equal to log(O(*a*)). If there is no path between *a* and *b*, then d_{ab} will be set a big value. In this method after the distance computation, pages are sorted in the ascending order and pages with smaller average distances will have high ranking.

This method is dependent on the out degree or out going links of nodes in the web graph like other algorithms. Apart from that it also follows the web graph like the random-surfer model [16] used in *PageRank* in that each output link of page *a* is selected with probability 1/O(*a*). That is rank's effect of *a* on page *b* as the inverse product of the out-degrees of pages in the logarithmic shortest path between *a* and *b*. For example, if there is the logarithmic shortest path with single length 3 from *a* to *b* like *a*→ *c* → *d* → *b*, then *a*'s effect on *b* is (1/O(*a*)) * (1/O(*c*)) * (1/O(*d*)) * (1/O(*b*)). In other words, the probability that a random surfer started from page *a* to reach to page *b* is (1/O(*a*)) * (1/O(*c*)) * (1/O(*d*)) * (1/O(*b*)).

According to the authors that if the distance between *a* and *b*, d_{ab} is less than the distance between *a* and *c*, d_{ac} then *a*'s rank effect, r_{ab} on *b* is more than on *c*, i.e if $d_{ab} < d_{ac}$ the $r_{ab} > r_{ac}$.. In other words, the probability that a random surfer reach *b* from *a* is more than the probability to reach from *c*.

The purpose of the *DistanceRank* is to compute the average distance of each page and there is a dependency between the distance of each page and its incoming links or back links. For example, if page *b* has only one back link and it is from page *a*, to compute the average distance for page *b*, d_b is as follows.

$$d_b = d_a + \log(\mathrm{O}(a)) \tag{22}$$

In general, suppose O(*a*) denotes the number of forwarding or outgoing links from page *a* and *B*(*b*) denotes the set of pages pointing to page *b*. The *DistanceRank* of page *b* denoted by d_b is given as follows.

$$d_b = \min(d_a + \log O(a)), a \in B(b) \tag{23}$$

The distance d_t from Fig. 8 is calculated as follows.

d_t = min{d_o + log2, d_p + log3} = min{d_m + log3 + log2, d_m + log3 + log3} = {d_m + log3 + log2} = d_m + 0.77.

According to the authors, the *DistanceRank* is similar to *PageRank* in ranking pages. Using (23), the authors proposed the following formula based on the Q-learning that is a type of reinforcement learning algorithm [24] to compute the distance of page *b* (*a* links to *b*).

$$d_{b_{t+1}} = (1-\alpha) * d_{b_t} + \alpha * \min(\log(O(a)) + \gamma^* d_{a_t}), a \in B(b), 0 < \alpha \le 1, 0 \le \gamma \le 1 \tag{24}$$

where α is learning rate and log(O(*a*) is the instantaneous punishment it receives in transition state from *a* to *b*. d_{b_t} and d_{a_t} show distance of page *b* and *a* in time *t* respectively and $d_{b_{t+1}}$ is distance of page *b* at time *t* + 1. In other words, the distance of page *b* at time *t* + 1 depends on its previous distance, its father distance (d_a) and log(*a*), the instantaneous punishment from selection page *b* by the user. The discount factor γ is used to regulate the effects of the distance of pages in the path leading to page *b* on the distance of page *b*. For example, if there is a path *m* → *n* → *o* → *p*, then the effect of the distance of *m* on *o* is regulated with a *γ* factor. In this fashion, the sum of received punishments is going to decrease. Since (24) is based on the reinforcement learning algorithm, it will converge finally and reach to the global optimum state [24].

The following (25) shows the learning rate α, where *t* shows time or iteration number and β is a static value to control regularity of the learning rate. If the learning rate is properly adjusted, the system will convergence and reach to the stability state very fast with a high throughput. In the beginning the distances of pages are not known, so initially α is set to one and then decreases exponentially to zero.

$$\alpha = e^{-\beta * t} \tag{25}$$

According to the authors, the user is an agent surfing the web randomly and in each step it receives some punishments from the environment. The goal is to minimize the sum of punishments. In each state, the agent has some selections, next pages for click, and the page with the minimum received punishment will be selected as the next page for visiting. With that the (24) can be modified as follows:

d_b = *α* * (previous punishment of selecting *b*) + (1- *α*) * (current punishment + instantaneous punishment that user will receive from selection *b*)

So d_b is the total punishment an agent receives from selection page *b*.

This system tries to simulate the real user surfing the web. When a user starts browsing a random page, he/she does not have any background about the web. Then, by browsing and visiting web pages, he/she clicks links based on both the current status of web pages and the previous experiences. As the time goes, the user gains knowledge in browsing and gets the favorite pages faster. *DistanceRank* uses the same kind of approach like a real user, sets initially α = 1 and after visiting more pages the system gets more information, α decreases and effectively selects the next pages.

The *DistanceRank* is computed recursively like *PageRank* as shown in (24). The process iterates to converge. It is possible [23] to compute distances with O(*p* * |*E*|) time complexity when $p << V$ which is very close to an ideal state. For instance, *p* is 7 for 7 millions pages implying that 7 iterations are enough for an acceptable ranking.

After convergence, the *DistanceRank* vector is produced. Pages with low *DistanceRank* will have high ranking and are sorted in the ascending order. The authors used two scenarios for experimental purpose. One is crawling scheduling and the other is rank ordering. The objective of the crawling scheduling is to find more important pages faster. In the rank ordering, *DistanceRank* is compared with *PageRank* and Google's rank with and without respect to a user query.

Based on the experimental results done by the authors, the crawling algorithms used by the *DistanceRank* outperforms [23] other algorithms like Breadth-first, Partial *PageRank*, Back-Link and OPIC in terms of throughput. That is *DistanceRank* finds high important pages faster than other algorithms. Also on the rank ordering, the *DistanceRank* was better than *PageRank* and Google's rank. The results of *DistanceRank* are closer to Google than *PageRank.*

DistanceRank and ranking problems

One of the main problems in the current search engines is "rich-get-richer" that causes the new high quality pages receives less popularity. To study more on this problem Cho et al [26] proposed two models on how users discovers new pages, Random-Surfer and Search-Dominant model. In Random-Surfer, a user finds new pages by surfing the web randomly without the help of search engines while Search-Dominant model finds new pages using search engines. The authors found out that it takes 60 times longer for a new page to become popular under Search-Dominant model than Random-Surfer model. If a ranking algorithm can find new high quality pages and increase their popularity earlier [25] then that algorithm is less sensitive to "rich-get-richer" problem. That is the algorithms should predict popularity that pages will get in the future.

This *DistanceRank* algorithm is less sensitive to "rich-get-richer" problem. *DistanceRank* algorithm also provides good prediction of pages for future ranking. The convergence speed of this algorithm is fast with a little number of iterations. In *DistanceRank,* it is not necessary to change the web graph for computation. Therefore some parameters like damping factor can be removed and can work on the real graph.

F. DirichletRank Algorithm

DirichletRank algorithm is proposed by Xuanhui Wang et al [27]. This algorithm eliminates the zero-one gap problem found in the *PageRank* algorithm proposed by Brin and Page [19]. The zero-one gap problem occurs due to the current ad hoc way of computing transition probabilities in the random surfing model. The authors proposed a novel *DirichletRank* algorithm which calculates the probabilities using the Bayesian estimation of Dirichlet prior. This zero-one gap problem can be exploited to spam *PageRank* results and make the state-of-art link-based anti spamming techniques ineffective. *DirichletRank* is a form of *PageRank* and the authors have shown that *DirichletRank* algorithm is free from the zero-one gap problem. They have also proved that this algorithm is more robust against several common link spams and is more stable under link perturbations. The authors also claim that this is as efficient as *PageRank* and it is scalable to large-scale web applications.

Search engines are getting more and more popular and it becomes the default method for information acquisition. Every body wants their pages to be on the top of the search result. This leads to the web spamming. Web spamming [28] is a method to maliciously induce bias to search engines so that certain target pages will be ranked much higher than they deserve. Consequently, it leads to poor quality of search results and in turn will reduce the trust of search engine.

Anti-spamming is now a big challenge for all the search engines. Earlier Web spamming was done by adding a variety of query keywords on page contents regardless of their relevance. This type of spamming is easy to detect and now the spammers are trying to use the link spamming [29] after the popularity of the link-based algorithm like *PageRank.* In link spamming, the spammers intentionally set up link structures, involving a lot of interconnected pages to boost the *PageRank* scores of a small number of target pages. This link spamming is not only increases rank gains but also harder to detect by the search engines. Fig. 9 (b) shows a sample link spam structure. Here, the leakage is used to refer to the *PageRank* scores that reach the link farm from external pages. In this, a web owner creates a large number of bogus web pages called *B's* (their sole purpose is to promote the target page's ranking score), all pointing to and pointed by a single target page *T*. The *PageRank* assigns a higher ranking score to *T* than it deserves (sometime up to 10 times of the original score) because it can be deceived by the link spamming.

Xuanhui Wang et al [27] proved that *PageRank* has a zero-one gap flaw which can be potentially exploited by spammers to easily spam *PageRank* results. This zero-one gap problem occurs from the ad hoc way of computing the transition probabilities in the random surfing model adopted in the current. The probability that the random surfer clicks on one link is solely given by the number of links on that page. This is why one page's *PageRank* is not completely passed on to a page it links to, but is divided by the number of links on the page. So, the probability for the random surfer reaching one page is the sum of probabilities for the random surfer following links to this page. Now, this probability is reduced by the damping factor *d*. The justification within the Random Surfer Model, therefore, is that the surfer does not click on an infinite number of links, but gets bored sometimes and jumps to another page at random. The zero-one gap problem refers to the unreasonable dramatic difference between a page with no out-link and one with a single out-link in their probabilities of randomly jumping to any page. The authors provided a novel

DirichletRank algorithm based on the Bayesian estimation with a Dirichlet prior to solve the zero-one gap problem specially the transition probabilities.

Zero-one gap problem

The basic *PageRank* assumes each row of the matrix M has at least one non-zero entry i.e. corresponding node in G has at least one out-link. But in reality it does not true. Many web pages does not have any out-links and many web applications only consider a sub-graph of the whole web. Even if a page has out-links it might have been removed when the whole web is projected to a sub-graph. Removing all the pages without out-links is not a solution because it generates new zero-out-link pages. This dangling page problem has been described by Brin and Page [16], Bianchini et al [30] and Ding et al [31]. The probability of jumping to random pages is 1 in zero-out-link page, but it drops to λ (in most cases, λ = 0.15) for a page with a single out-link. There is a big difference between 0 and 1 out-link. This problem is referred as "zero-one gap". This problem is a serious flaw in the *PageRank* because it allows spammers to manipulate the ranking results of *PageRank*.

Fig. 9 (a) is a structure without link spamming (only out-link) and Fig. 9 (b) shows a typical spamming structure with all bogus pages *B*'s having back links to the target page *T*. The authors denote $r_o(.)$ as the *PageRank* score in Fig.9 (a) and $r_s(.)$ denotes the *PageRank* score in Fig. 9 (b). The authors proved that $r_s(T) \geq r_o(T)$ over the range of all λ values. Usually a small λ is preferred in *PageRank* so the result in $r_s(T)$ is much larger than $r_o(T)$. For example if λ = 0.15, $r_s(T)$ is about 3 times larger than $r_o(T)$. Fig. 9 (b), the addition of the bogus pages makes the *PageRank* score of the target page 3 times larger than before. This is because a surfer is forced to jump back to the target page with a high probability in Fig 9 (b). With the default value of λ = 0.15, the single out-link in a bogus page forces a surfer to jump back to the target page with a probability 0.85. This zero-one-gap problem denotes a serious flaw of *PageRank*, which makes it sensitive to a local structure change and thus vulnerable to link spamming.

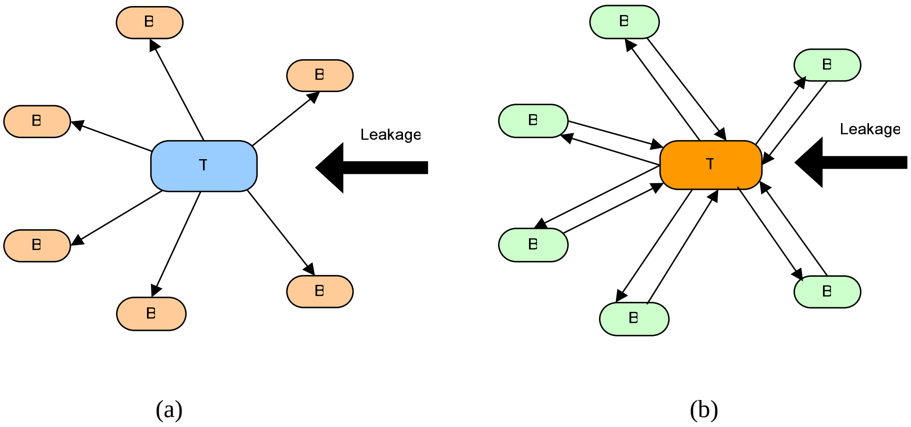


Fig. 9 Sample contrast structures

DirichletRank Algorithm

DirichletRank algorithm based on the Bayesian estimation of transition probabilities. According to the authors this algorithm not only solves the zero-one gap problem but also provides a way to solve the zero-out-link problem. The authors compared *DirichletRank* with *PageRank* and showed that *DirichletRank* is less sensitive to the changes in the local structure and more robust than *PageRank*.

In *DirichletRank*, a surfer would more likely follow the out-links of the current page if the page has many out-links. Bayesian estimation provides a proper way for setting the transition probabilities and the authors showed that it is not only solves the zero-out-link problem but also solves the zero-one gap problem. The random jumping probability of *DirichletRank* is

$$\omega(n) = \frac{\mu}{n+\mu}, 0 \leq n \leq \infty \qquad (26)$$

where n is the number of out-links and μ is the Dirichlet parameter. The author set μ = 20 and plotted $\omega(n)$ and showed the jumping probability in *DirichletRank* is smoothed and no gap between 0 and 1 out-link. The authors calculated *DirichletRank* scores $d_o(.)$ and $d_s(.)$ for the structures shown in Fig. 9 (a) and (b) using the following formula.

$$d_o(T) = \sigma + \frac{\tau}{N} \qquad (27)$$

$$d_s(T) = \left[1 + \frac{k}{\mu^2 + (k+1)\mu}\right]\left[\sigma + \frac{k+\mu+1}{\mu+1}\right]\frac{\tau}{N} \qquad (28)$$

and $d_s(T) \geq d_o(T)$ for any positive integer *k*.

The authors claimed that they obtained a similar score of *PageRank* i.e. $d_s(T)$ is constantly larger than or equal to $d_o(T)$ but $d_s(T)$ is in fact close to $d_o(T)$. It also shows that no significant change in *T*'s *DirichletRank* scores before and after spamming. Hence *DirichletRank* is more stable and less sensitive to the change of local structure. *DirichletRank* also don't take extra time cost. And it makes suitable for Web-scale applications. The authors also proved that *DirichletRank* is more stable than *PageRank* during link perturbation i.e. removing a small number of links or pages. Stability is an important factor for a reliable ranking algorithm. Their experiment results showed that *DirichletRank* is more effective than *PageRank* due to its more reasonable allocation of transition probabilities.

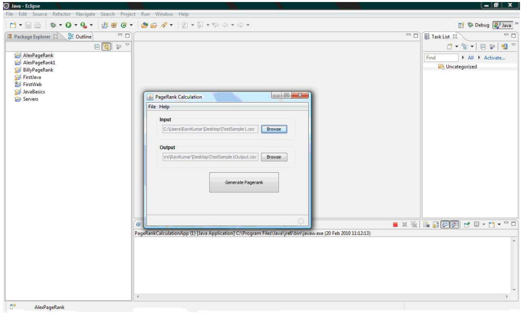
Table II shows the comparison [2] of all the algorithms discussed above. The main criteria used for comparison are mining techniques used, working method, input parameters, complexity, limitations and the search engine using the algorithm. Among all the algorithms, PageRanka and HITS are most important ones. PageRank is the only algorithm implemented in the Google search engine. HITS is used in the IBM prototype search engine called Clever. A similar algorithm is used in the Teoma search engine and later it is acquired by Ask.com. HITS can not be implemented directly in a search engine due to its topic drift and efficiency problem. That is the reason we have taken PageRank algorithm and implemented in a Java program.

TABLE II
COMPARISON OF PAGERANK BASED ALGORITHMS

Algorithm Criteria	*PageRank*	*Weighted PageRank*	*HITS*	*Distance Rank*	*Dirichlet Rank*
Mining technique used	*WSM*	*WSM*	*WSM & WCM*	*WSM*	*WSM*
Working	Computes scores at index time. Results are sorted on the importance of pages.	Computes scores at index time. Results are sorted on the Page importance.	Computes scores of n highly relevant pages on the fly.	Computes scores by calculating the minimum average distance between pages	Works same as PageRank but computes transition probabilities using Bayesian estimation
I/P Parameters	Backlinks	Backlinks, Forward links	Backlinks, Forward Links & content	Backlinks	Backlinks
Complexity	O(log N)	<O(log N)	<O(log N)	O(log N)	O(log N)
Limitations	Query independent	Query independent	Topic drift and efficiency problem	Needs to work along with PageRank	Needs to work along with PageRank
Search Engine	Google	Research model	Clever	Research Model	Research Model

IV SIMULATION RESULTS

The program was developed for the *PageRank* algorithm using Java and tested on an Intel Core (2 duo) with 4GB RAM machine. The input is shown in Fig. 10, the user can select the input file which contains the number of nodes, the number of incoming and outgoing links of the nodes. The output is shown in Fig. 11 (a) and (b). Fig. 11 (a) is the output of the *PageRank* convergence scores and Fig. 11 (b) is the XY chart for the *PageRank* scores.



C:\Users\RaviKumar\Desktop\TestSample1.csv

ers\RaviKumar\Desktop\TestSample1Output.csv

PageRankCalculationApp (1) [Java Application] C:\Program Files\Java\jre6\bin\javaw.exe (20 Feb 2010 11:12:13)

Fig. 10 PageRank Program Input Entry Window

Iteration	A	B	C	D
1	1	1	1	1
2	1.566667	1.099167	1.127264	0.780822
3	1.444521	1.083313	1.07086	0.760349
4	1.406645	1.051235	1.045674	0.744124
..	..	..	..	..
..	..	..	..	..
33	1.31351	0.988244	0.988244	0.710005
34	1.313509	0.988244	0.988244	0.710005
35	1.313509	0.988244	0.988244	0.710005

Fig. 11 (a) PageRank Convergence Scores

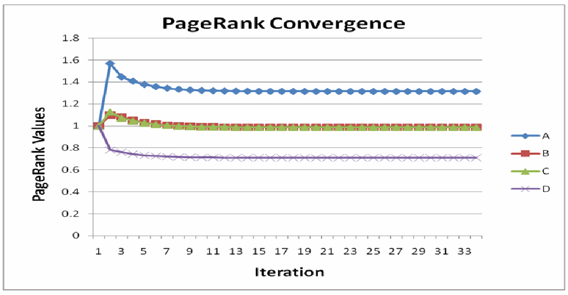


Fig. 11 (b) PageRank Convergence Chart

V Conclusion

This paper covers the basics of Web mining and the three areas of Web mining used for Information Retrieval. Web Structure mining and Web Graph are explained in detail to have a better understanding of the data structure used in web. The main purpose of this paper is to explore the important Page Rank based algorithms used for information retrieval and compare those algorithms. The future work will be apply the PageRank program in the Web and compare it with the original PageRank algorithm Finally, simulation results are shown for the *PageRank* algorithm.

Acknowledgements

Authors would like to acknowledge Alex Goh Kwang Leng and Billy Lau Pik Lik, Computer Science students of Curtin University of Technology, Sarawak for their contribution in the simulation program.

References

[1] M. G. da Gomes Jr. and Z.Gong, “Web Structure Mining: An Introduction”, *Proceedings of the IEEE International Conference on Information Acquisition*, 2005.

[2] N. Duhan, A. K. Sharma and K. K. Bhatia, “Page Ranking Algorithms: A Survey, *Proceedings of the IEEE International Conference on Advance Computing*, 2009.

[3] R. Kosala, H. Blockeel, “Web Mining Research: A Survey”, *SIGKDD Explorations, Newsletter of the ACM Special Interest Group on Knowledge Discovery and Data Mining* Vol. 2, No. 1 pp 1-15, 2000.

[4] R. Cooley, B. Mobasher and J. Srivastava, “Web Minig: Information and Pattern Discovery on the World Wide Web”. *Proceedings of the 9th IEEE International Conference on Tools with Artificial Intelligence*, pp. (ICTAI’97), 1997.

[5] http://googleblog.blogspot.com/2008/07.

[6] E. Horowitz, S. Sahni and S. Rajasekaran, “Fundamentals of Computer Algorithms”, *Galgotia Publications Pvt. Ltd*., pp. 112-118, 2008.

[7] A. Broder, R. Kumar, F Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, J. Wiener, “Graph Structure in the Web”, *Computer Networks: The International Journal of Computer and telecommunications Networking*, Vol. 33, Issue 1-6, pp 309-320, 2000.

[8] R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tompkins and E. Upfal, “Web as a Graph”, *Proceedings of the Nineteenth ACM SIGMOD-SIGACT-SIGART symposium on Database systems*, 2000.

[9] J. Kleinberg, R. Kumar, P. Raghavan, P. Rajagopalan and A. Tompkins, “Web as a Graph: Measurements, models and methods”, *Proceedings of the International Conference on Combinatorics and Computing*, pp. 1-18, 1999.

[10] E. Garfield, “Citation Analysis as a tool in journal evaluation”, *Science* 178, pp. 471-479, 1972.

[11] 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*, 1976.

[12] D. Gibson, J. Kleinberg, P. Raghavan, “Inferring Web Communities from Link Topology”, *Proc. of the 9th ACM Conference on Hypertext and Hypermedia*, 1998.

[13] R. Kumar, P .Raghavan, S .Rajagopalan, A. Tomkins, “Trawling the Web for Emerging Cyber-Communities”, *Proc. of the 8th WWW Conference (WWW8)*, 1999.

[14] J. Dean and M. Henzinger, “Finding Related Pages in the World Wide Web”, *Proc. Eight Int’l World Wide Web Conf*., pp. 389-401, 1999.

[15] J. Hou and Y. Zhang, “Effectively Finding Relevant Web Pages from Linkage Information”, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 15, No. 4, 2003.

[16] S. Brin, L. Page, “The Anatomy of a Large Scale Hypertextual Web search engine,” *Computer Network and ISDN Systems*, Vol. 30, Issue 1-7, pp. 107-117, 1998.

[17] W. Xing and Ali Ghorbani, “Weighted PageRank Algorithm”, *Proc. of the Second Annual Conference on Communication Networks and Services Research (CNSR ’*04), *IEEE*, 2004.

[18] J. Kleinberg, “Authoritative Sources in a Hyper-Linked Environment”, *Journal of the ACM* 46(5), pp. 604-632, 1999.

[19] L. Page, S. Brin, R. Motwani, and T. Winograd, “The Pagerank Citation Ranking: Bringing order to the Web”. *Technical Report, Stanford Digital Libraries* SIDL-WP-1999-0120, 1999.

[20] C. Ridings and M. Shishigin, “PageRank Convered”. *Technical Report*, 2002.

[21] J. Kleinberg, “Hubs, Authorities and Communities”, *ACM Computing Surveys*, 31(4), 1999.

[22] S. Chakrabarti, B. Dom, D. Gibson, J. Kleinberg, R. Kumar, P. Raghavan, S. Rajagopalan, A. Tomkins, “Mining the Link Structure of the World Wide Web”, *IEEE Computer Society Press*, Vol 32, Issue 8 pp. 60 – 67, 1999.

[23] A. M. Zareh Bidoki and N. Yazdani, “DistanceRank: An intelligent ranking algorithm for web pages” *Information Processing and Management*, Vol 44, No. 2, pp. 877-892, 2008.

[24] R.S. Sutton and A.G. Barto, “Reinforcement Learning: An Introduction”. Cambridge, MA: MIT Press, 1998

[25] J. Cho, S. Roy and R. E. Adams, “Page Quality: In search of an unbiased web ranking”. *Proc. of ACM International Conference on Management of Data”*. Pp. 551-562, 2005.

[26] J. Cho and S. Roy, “Impact of Search Engines on Page Popularity”. *Proc. of the 13th International Conference on WWW*, pp. 20-29, 2004.

[27] X. Wang, T. Tao, J. T. Sun, A. Shakery and C. Zhai, “DirichletRank: Solving the Zero-One Gap Problem of PageRank”. *ACM Transaction on Information Systems*, Vol. 26, Issue 2, 2008.

[28] Z. Gyongyi and H. Garcia-Molina, “Web Spam Taxonomy”. *Proc. of the First International Workshop on Adversarial Information Retrieval on the Web”*, 2005.

[29] Z.. Gyongyi and H. Garcia-Molina, “Link Spam Alliances”. *Proc. of the 31st International Conference on Very Large DataBases (VLDB)*, pp. 517-528, 2005.

[30] M. Bianchini, M.. Gori and F. Scarselli, “Inside PageRank”. *ACM Transactions on Internet Technology*, Vol. 5, Issue 1, 2005

[31] C.. H. Q. Ding, X. He, P. Husbands, H. Zha and H. D. Simon, “PageRank: HITS and a Unified Framework for Link Analysis”. *Proc. of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2002.

[32] http://toolbar.google.com/.