Random Web Surfer PageRank Algorithm

Navadiya Hareshkumar, Dr. Deepak Garg
Computer Science and Engineering Department
Thapar University, Punjab

ABSTRACT
In this paper analyzes how the Google web search engine implements the PageRank algorithm to define prominent status to web pages in a network. It describes the PageRank algorithm as a Markov process, web page as state of Markov chain, Link structure of web as Transitions probability matrix of Markov chains, the solution to an eigenvector equation and Vector iteration power method.

It mainly focus on how to relate the eigenvalues and eigenvector of Google matrix to PageRank values to guarantee that there is a single stationary distribution vector to which the PageRank algorithm converges and efficiently compute the PageRank for large sets of web Pages. Finally, it will demonstrate example of the PageRank algorithm.

Keywords
PageRank, Markov chains, Power method, Google matrix, stationary distribution vector, Eigen Vector and Values

1. INTRODUCTION
Paper begins with a review of the most basic PageRank model for determining the importance of a webpage.

Paper provides comprehensive survey of all issues associated with PageRank, available and recommended solution methods, changing α value and convergence rate, storage issues and compares PageRank to an idealized random Web surfer and shows how to efficiently compute PageRank for large numbers of pages.

2. PAGERANK
A search query with Google's search engine usually returns a very large number of pages. E.g., a search on 'University' returns 225 million pages. Although the search returns several million pages, the most relevant pages are usually found within the top ten or twenty pages in the list of results. How does the search engine know which pages are the most important?

Google assigns a number to each individual webpage based on the link structure of the web, expressing its importance. This number is known as the PageRank and is computed via the PageRank Algorithm. PageRank has applications in search, browsing, and traffic estimation.

3. BASIC PAGERANK ALGORITHM MODEL
An intuitive definition of PageRank algorithm shows how one might measure the importance of a web page on the Internet.

A webpage **U**'s PageRank is calculated base on how many other WebPages Backlink (inedges) into **U**. The PageRank of **U** is the sum of the PageRanks of each webpage $\mathbf{V_i}$ that backlinks to **U** divided by the number of WebPages to which $\mathbf{V_i}$ links [1]. This intuitive definition says that: if webpage **U** is linked to only by WebPages with low PageRanks, then **U** may not get more importance. Further, if **U** is linked to by a page $\mathbf{V_j}$ with a high PageRank, but $\mathbf{V_j}$ links to many other pages, **U** should not receive the full weight (benefits) of $\mathbf{V_j}$'s PageRank.

For example, if a webpage has a link to the **Facebook Home** page, it may be just one link but it is a very important one. This page should be ranked higher than many pages with more links but from lower important pages. PageRank is an attempt to see how good an approximation to "importance" can be obtained from the link structure.

3.1 FORMAL DEFINITION
Now that we have an idea of how PageRank works, we will formally define the algorithm. Drawing on [2], we define PageRank as follows:

- Let U be a web page.
- Let F_U be the set of Forward links (outedges) from U.
- Let B_U be the set of Backlinks into U.
- Let $N_U = |F_U|$ be the number of forward links from U.
- Let c be a normalization factor so that "the total rank of all web pages is constant" [2].

Fig 1: A Webpage's Link Structure

We begin by defining a simple ranking, R which is a slightly simplified version of PageRank:

$$R(U) = c \sum_{V \epsilon B_U} \frac{R(V)}{N_V} \qquad (1)$$

Note that here $c < 1$ because there are number of pages with no forward links and their weight is lost from the system (see **Section 4.2**). The equation is recursive but it may be computed by starting with any set of ranks and iterating the computation until it converges. **Fig 2** demonstrates the propagation of rank from one pair of pages to another.

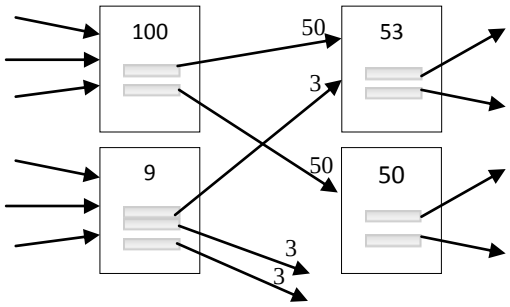

Fig 2: Simplified PageRank Calculation

There is a small problem with this simplified ranking algorithm. Consider two web pages that point to each other but to no other page. And suppose there is some web page which points to one of them. Then, during iteration, this loop will accumulate rank but never distribute any rank (since there are no outedges). The loop forms a sort of trap which is called **a Rank Sink.**

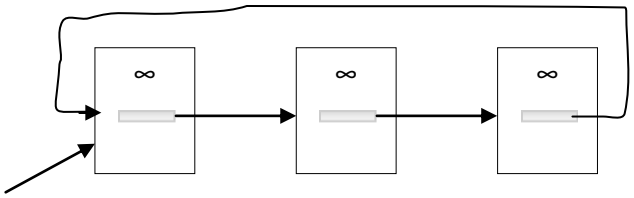

Fig 3: Loop which acts as a Rank Sink

To overcome this problem of rank sinks, we introduce a rank source:

- Let E(U) be "some vector over the Web pages that corresponds to a source of rank" [2].

Then, the PageRank of page U is given by

$$R'(U) = c \sum_{V \epsilon B_U} \frac{R'(V)}{N_V} + c\, E(U) \qquad (2)$$

This is iterated until the values have converged sufficiently [3].

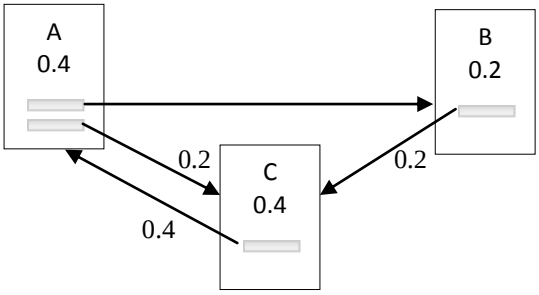

Fig4: Simplified PageRank Calculation

Notice that PageRank can be described using **equation 2**, the summation method is neither the most interesting nor the most illustrative of the algorithm's properties. So, it is preferred to rewrite the sum as Matrix Multiplication.

4. RANDOM WEB SURFER MODEL

PageRank also has a second definition based upon the model of a random web surfer navigating the Internet. In short, the model states that PageRank models the behavior of someone who "keeps clicking on successive links at random". However, occasionally the surfer "gets bored and jumps to a random page chosen based on the distribution in E." [2]. so, surfer will not continue in the loop forever.

Consider the illustration in **Fig 5** of a simple Link Structure of web pages; we can represent this structure using an N × N **adjacency matrix** A, where $\mathbf{A_{ij} = 1}$ if there is a link (Edge) from webpage (Vertex) i to webpage j, and 0 otherwise [4].

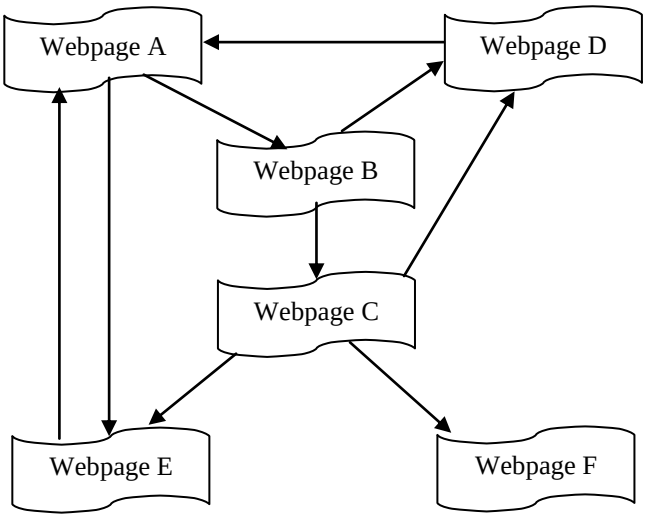

Fig 5: A Simple Link Structure

Thus, the 6×6 adjacency matrix for the link structure in **Fig 5** is

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Notice that the diagonal entries are all zero. We assume that links from a page to itself are ignored when constructing an adjacency matrix.

To illustrate the PageRank algorithm based on second definition, the following variables are defined:

- Let N be the total number of web pages in the web.
- Let π^T be the 1×N PageRank row vector (Stationary vector).
- Let H the N × N **row-normalized** adjacency matrix. (Transition Probability Matrix)

Then we can describe the PageRank vector at the k^{th} iteration as

$$\pi^{(k)T} = \pi^{(k-1)T} H \qquad (3)$$

So that successive iterations $\pi^{(k)T}$ converge to the PageRank vector π^T [3].

4.1 ROW-NORMALITY

Let to build a Transition Probabilities matrix,

$$H_i = \frac{A_i}{\sum_{k=1}^{N} A_{ik}} \qquad (4)$$

So, that each row A_i of A is divided by its row sum. Apply **equation 4** on Adjacency matrix A. so we have, sparse matrix

$$\mathbf{H} = \begin{pmatrix} 0 & 1/2 & 0 & 0 & 1/2 & 0 \\ 0 & 0 & 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 1/3 & 1/3 & 1/3 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Matrix **H** whose element H_{ij} is the probability of moving from state i (page i) to state j (page j) in one time step (In **Fig 5**, assume that, starting from any webpage, it is Half the probability to arrive from page A to page B in one time step).

4.2 DANGLING LINKS

One issue with this model is dangling links. Dangling links are simply links that point to any page with no outgoing links (In **Fig 5**, Link to Webpage F) and Pages with no outlinks are called **dangling nodes** (Webpage F). They affect the model because it is not clear where their weight should be distributed. A page of data, a page with a postscript graph, a page with jpeg pictures, a pdf document, a page that has been fetched by a crawler but not yet explored - these are all examples of possible dangling nodes. In fact, for some subsets of the web, dangling nodes make up 80% of the collection's pages [5].

4.3 STOCHASTICITY

Using the matrix H is insufficient for the PageRank algorithm, however, because the iteration using H alone might not converge properly — "it may be dependent on the startingvector"[3]. Here matrix H is not yet **stochastic** [3]. A matrix is stochastic when it is a "matrix of transition probabilities for a **Markov chain,**" **[**A Markov chain is "collection of random variables" whose future values are independent of their past values [6].**]** with the property that "all elements are non-negative and all its row sums are unity" (one) [7]. Thus, to ensure that H is stochastic, we must ensure that every row sums to one. But here, the sum of the last row of matrix H is Zero because of **Dangling Node** (see **Section 4.2**).

To overcome this problem, pages with no Forward links (**Webpage F**) are assigned artificial links or "teleporters" to all other pages.

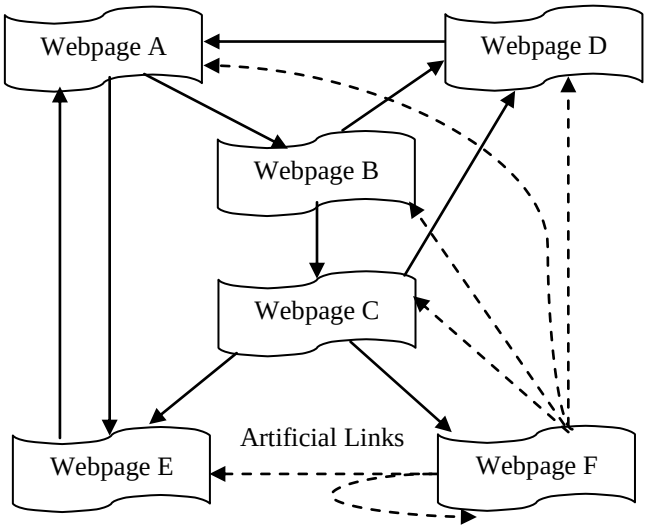

Fig 6: A StochasticLink Structure

Therefore, we define the stochastic S as,

$$S = H + \frac{a * e^T}{N} \qquad (5)$$

Where,
a = N×1 Column Vector such that
$a_i = 1$ if $\sum_{k=1}^{N} H_{ik} = 0$ (i.e. Page i is Dangling Page)
$= 0$ otherwise
e = N×1 Column Vector of one's [3].

Apply **equation 5** on matrix H. so we have stochastic matrix S as,

$$\mathbf{S} = \begin{pmatrix} 0 & 1/2 & 0 & 0 & 1/2 & 0 \\ 0 & 0 & 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 1/3 & 1/3 & 1/3 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \end{pmatrix}$$

It makes sure that surfer's random walk process does not get stuck and the web pages are the states of the Markov chain.

4.4 THE GOOGLE MATRIX

There is no guarantee that S has a unique **stationary distribution vector** (i.e., there might not be a single "correct" PageRank vector) [3]. For us to guarantee that there is a single stationary distribution vector π^T to which we can converge, we must ensure that S is **irreducible** as well as stochastic [3]. A matrix is irreducible if and only if its graph is **strongly connected** [8].

So we define the irreducible row-stochastic matrix G as

$$G = \alpha S + (1-\alpha)E \;;\; 0 \le \alpha \le 1 \qquad (6)$$
$$E = \frac{e * e^T}{N}$$

G is the Google matrix, and we can use it to define

$$\pi^{(k)T} = \pi^{(k-1)T} G \qquad (7)$$

as the new iterative method for PageRank to replace **equation 3**. Google uses the value of 0.85 for α [3] (see S**ection 6**), and we will use this same value. Thus, apply **equation6** on matrix S, our G is, **G =**

$$\begin{pmatrix} 1/40 & 9/20 & 1/40 & 1/40 & 9/20 & 1/40 \\ 1/40 & 1/40 & 9/20 & 9/20 & 1/40 & 1/40 \\ 1/40 & 1/40 & 1/40 & 77/250 & 77/250 & 77/250 \\ 7/8 & 1/40 & 1/40 & 1/40 & 1/40 & 1/40 \\ 7/8 & 1/40 & 1/40 & 1/40 & 1/40 & 1/40 \\ 83/500 & 83/500 & 83/500 & 83/500 & 83/500 & 83/500 \end{pmatrix}$$

Since the matrix G is completely dense, and the graph of G now strongly connected and G is irreducible.

5. THE POWER METHOD

The Google matrix G is currently of size more than eight billion web pages [9] and therefore the Eigen value computation is not trivial. To find an approximation of the principal Eigenvector the **Power Method**is used.

We iterate using the Google matrix G by writing **equation 7**. Let us take 1×N initial row vector$\pi^{(0)T} = \frac{e^T}{N}$ Based on it result found is, $\boldsymbol{\pi^{(19)T}} =$

$$(0.320 \quad 0.170 \quad 0.107 \quad 0.137 \quad 0.200 \quad 0.064)$$

The PageRank of page i is given by the i[th] element of π^T.

Table 1.PageRank of Tiny 6 web pages Link Structure

Web Page	PageRank	Relative importance
A	0.320	1
B	0.170	3
C	0.107	5
D	0.137	4
E	0.200	2
F	0.064	6
	Total = 1	

Starting anywhere within **Fig 5** and randomly surfing the web pages in given network, the likelihood of ending up at any of the Pages are, respectively, about 32%, 17%, 11%, 14%, 20 % and 6%.

5.1 IMPROVEMENT OF POWER METHOD

When dealing with large data sets, it is difficult to form a matrix G and find its dominant eigenvector. It is more efficient to compute the PageRank vector using the power method, where we iterate using the **sparse matrix H** by rewriting **equation 7**.

$$\begin{aligned} \pi^{(k)T} &= \pi^{(k-1)T} G \\ &= \pi^{(k-1)T}(\alpha S + (1-\alpha)E) \\ &= \pi^{(k-1)T}\left(\alpha S + (1-\alpha)\frac{e * e^T}{N}\right) \\ =&\alpha\pi^{(k-1)T} S + (1-\alpha)\pi^{(k-1)T}\frac{e*e^T}{N} \\ =&\alpha\pi^{(k-1)T} S + (1-\alpha)\frac{e^T}{N} \\ =&\alpha\pi^{(k-1)T}\left(H + \frac{a*e^T}{N}\right) + (1-\alpha)\frac{e^T}{N} \\ =&\alpha\pi^{(k-1)T} H + \left(\alpha\pi^{(k-1)T} a + (1-\alpha)\right)\frac{e^T}{N} \end{aligned} \qquad (8)$$

Since $\pi^{(k-1)T}$ is a probability vector, and thus, $\pi^{(k-1)T} e = 1$ Fortunately, since **H** is sparse, each vector-matrix multiplication required by the power method can be computed in *nnz*(**H**) flops, where *nnz*(**H**) is the number of nonzeros in **H** and since the average number of nonzeros per row in **H** is 3-10, $O(nnz(\mathbf{H})) \approx O(n)$ [5]. Furthermore, at each iterationthe power method only requires the storage of one vector, the current iterate. The founders of Google, Lawrence Page and Sergey Brin, use α = 0.85 and report success using only 50 to 100 power iterations [2].

6. CHANGING α VALUE ANDCONVERGENCE RATE

One of the most obvious places to begin fiddling with the basic PageRank model is α. Brin and Page, have reported using α = 0.85. One wonders why this choice for α? Might a different choice produce a very different ranking of retrieved WebPages? As mentioned in sections The Google Matrix and The Power Method, there are good reasons for using α = 0.85, one being the speedy convergence of the power method. With this value for *α*, rough estimate of the number of iterations needed to converge to a tolerance level **τ** (measured by the residual,$\pi^{(k)T} - \pi^{(k-1)T}$) is $\boldsymbol{log_{10}\tau}$ / $\boldsymbol{log_{10}\alpha}$. For **τ** = 10^{-6} and α = 0.85, one can expect roughly -6 / $log_{10}0.85 \approx 85$ iterations until convergence to the PageRank vector. For **τ** = 10^{-8}, about 114 iterations and for **τ** = 10^{-10}, about 142 iterations. Brin and Page report success using only 50 to 100 power iterations, implying that **τ** could range from 10^{-3} to 10^{-7}. Obviously, this choice of α brings faster convergence than higher values of α.

Compare with α = 0.99, whereby roughly 1833 iterations are required to achieve a residual less than 10^{-8}. When working with a sparse 8 billion by 8 billion matrix, each iteration counts; over a few hundred power iterations is more than Google is willing to compute. However, in addition to the computational reasons for choosing α = 0.85, this choice for *α* also carries some intuitive weight: α = 0.85 implies that roughly **5/6** of the time a Web surfer randomly clicks on hyperlinks (i.e., following the structure of the Web, as captured by the **α*S*** part of the **equation 6**), while **1/6** of the time this Web surfer will go to the URL line and type the address of a new page to "Artificial Link" to (as captured by the

$(\mathbf{1-\alpha})\mathbf{E}$ part of the **equation 6**). Perhaps this was the original motivation behind Brin and Page's choice of α = 0.85; it produces an accurate model for Web surfing behavior. Whereas α = 0.99, not only slows convergence of the power method, but also places much greater emphasis on the link structure of the Web and much less on the teleportation tendencies of surfers. The PageRank vector derived from α = 0.99 can be vastly different from that obtained using α = 0.85. Perhaps it gives a "truer" PageRanking. An Experiment with various α's show significant variation in rankings produced by different values of α[10].

7. ALTERNATIVE OF POWER METHOD

The iterative aggregation/disaggregation (IAD) method: is an improvement of the PageRank algorithm used bythe search engine Google to compute stationary probabilities of very large Markov chains.

It is shown that the power method applied to the Google matrix always converges, and that the convergence rate of theIAD method is at least as good as that of the power method. Furthermore, by exploiting the hyperlink structure of the web it can be shown that the convergence rate of the IAD method applied to the Google matrix can be made strictly faster than that of the power method [11].

The adaptive randomized method: for finding eigenvectorcorresponding to Eigen Value 1 for stochastic matrices hasbeen proposed. The upper bound of its rate of convergenceis of non-asymptotic type and has the explicit factor. Moreover, the bound is valid for the whole class of stochastic matrices and does not depend on propertiesof the individual matrix. The method can be applied for PageRank computation with small parameter α. Further workon acceleration of the method is possible [12].

DYNA-RANK: focuses upon efficiently calculating and updating Google's PageRank vector using "peer to peer" system. The changes inthe web structure will be handled incrementally amongstthe peers. DYNA-RANK produces the relative PageRank oneach peer. DYNA-RANK is proven to take less computation time and less number of iterations compared to centralized approach [13].

8. STORAGE ISSUES OF GOOGLE MATRIX

The size of the Markov matrix makes storage issues nontrivial. For subsets of the web, the transition probability matrix **H** may or may not fit in main memory. When a large matrix exceeds a machine's memory, researchers usually try one of two things:

1. Compress the data needed so that the compressed representation fits in main memory and then creatively implement a modified version of PageRank on this compressed representation.

2. Keep the data in its uncompressed form and develop I/O-efficient implementations of the computations that must take place on the large, uncompressed data.

For modern web structure for which the transition probability matrix **H** can be stored in main memory, compression of the data is not essential.

Rather than storing the full matrix or a compressed version of the matrix, Web-sized implementations of the PageRank model store the **H** or **A** matrix in an adjacency list of the columns of the matrix [14]. In order to compute the PageRank vector, the PageRank power method requires vector-matrix multiplications of $\boldsymbol{\pi^{(k-1)T}}\mathbf{H}$ at each iteration k. Therefore, quick access to the columns of the matrix **H** (or **A**) is essential to algorithm speed. Column i contains the inlink (Backlink) information for page i, which, for the PageRank system of ranking WebPages, is more important than outlink (Forwardlink) information contained in the rows of **H** or **A**. For the tiny 6-node web from **Fig 5**, an **adjacency list** representation of the columns of matrix **A** is:

Table 2.Adjacency list of matrix A

Web Page	BackLink From
A	D, E
B	A
C	B
D	B, C
E	A, C
F	C

Cleve Moler gives one possible implementation of the power method applied to an adjacency list, along with sample Matlab code [15]. When the adjacency list does not fit in main memory [14], suggest methods for compressing the data. Reference [16] take the other approach and suggest I/O-efficient implementations of PageRank. Since the PageRank vector itself is large and completely dense, containing over 8 billion pages, and must be consulted in order to process each user query, Reference [16] has also suggested a technique to compress the PageRank vector. This encoding of the PageRank vector hopes to keep the ranking information cached in main memory, thus speeding query processing.

9. CONCLUSION

In this paper, we have taken on the audacious task of condensing every page on the World Wide Web into a single number, its PageRank. PageRank is a global ranking of all web pages, regardless of their content, based solely on their location in the Web's link structure. Using PageRank, we are able to order search results so that more important and central WebPages are given preference.

The intuition behind Page Rank is that it uses information which is external to the Web pages themselves – their backlinks, which provide a kind of peer review. Furthermore, backlinks from "important" pages are more significant than backlinks from average pages. This is encompassed in the recursive definition of PageRank (**equation 2**).

In the proposed algorithm a value of α is usedthat play a very important role on the analysis of thealgorithm. After the analysis it is concluded that α must not be selected closer to zero. At α = 1, the system enters into the ideal state and theranking provided is insignificant. As per evaluation α must be selected greater than or equals to 0.5. However, if we consider convergence speed as only factor toevaluate the performance than the best factor α will be .85. The proposed algorithm is query independent algorithmand does not consider query during ranking.As the Web continues its amazing growth, the need for smarter storage schemes and evenfaster numerical methods will become more evident. Both are exciting areas for computer scientists and numerical analysts interested in information retrieval.

As future work, it is desirable to conduct more experiments using larger and various kinds of datasets in order to further validate our conclusions in this paper.

10. REFERENCES

[1] Desmond J. Higham and Alan Taylor, *The Sleekest Link Algorithm* (2003).

[2] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd, *The PageRank Citation Ranking: Bringing Order to the Web* (1998).

[3] Amy N. Langville and Carl D. Meyer, *The Use of the Linear Algebra by Web Search Engines* (2004).

[4] Eric W. Weisstein, *Adjacency Matrix*, From MathWorld- A Wolfram Web Resource.

[5] Amy N. Langville, Carl D. Meyer, *Deeper InsidePageRank*(2004).

[6] Eric W. Weisstein, *Markov Chain*, From MathWorld– A Wolfram Web Resource.

[7] David Nelson, editor, *The Penguin Dictionary of Mathematics* (Penguin Books Ltd, London, 2003).

[8] Sergio S. Guirreri, Markov Chains as methodology used byPageRank to rank the Web Pages on Internet (2010).

[9] Bill Coughran, *Google's index nearly doubles*, Google Inc.(2004)

[10] Kristen Thorson. *Modeling the Web and the computation of PageRank* (Hollins University, 2004).

[11] Ilse C.F. Ipsen, Steve Kirkland, *Convergence Analysis Of An Improved PageRank Algorithm* (2003)

[12] Alexander Nazin, Boris Polyak, Adaptive Randomized Algorithm for Finding Eigenvector of Stochastic Matrix with Application to PageRank (48th IEEE Conference- December 16-18, 2009)

[13] MandarKale, Mrs.P.SanthiThilagam, *DYNA-RANK: Efficient calculation and updation of PageRank*(International Conference on Computer Science and Information Technology 2008)

[14] Sriram Raghavan, Hector Garcia-Molina. *Compressing the graph structure of the Web*. In Proceedings of the IEEE Conference on Data Compression, pages 213–222, (March 2001).

[15] Cleve B. Moler. Numerical Computing with MATLAB. (SIAM, 2004).

[16] Taher H. Haveliwala. *Efficient encodings for documentranking vectors*. (Technical report, CS Department,Stanford University, November 2002).