PageRank

Manvi Gupta* (B17092), Indian Institute of Technology Mandi

***Abstract*—It was in 1998, that two doctorate students of Stanford, Larry Page, and Sergey Brin deployed their web searching algorithm PageRank for the first time. And it knocked down all the existing search algorithms present at that time. It brought Google where it is today, replacing Yahoo and Bing. It is a function that calculates a score for each web page using its outgoing and incoming links and ranks them accordingly. There was nothing as efficient as the PageRank. The simplicity of the famous algorithm, coupled with its efficient working, makes it stand out even today, despite the fact that the actual implementation and logic is still not fully disclosed to the public. The primary objective of this paper is to talk, in detail, about the working of PageRank, issues that were faced during its implementation and the scope of improvement.**

***Index Terms*—PageRank, HITS, Google, Search Engine, Greedy Algorithm**

I. Introduction

As we walked into the 21st century, a big transformation took place in web search technology. Though many search engines existed before Google, it was able to defeat all. Such a revolutionary change was brought in by "PageRank", a web search algorithm developed by *Larry Page* and *Sergey Brin*, better known as *the founders of Google*. It made web search simpler and more efficient than all other existing counterparts. Yet, there were some spammers who tried to manipulate the PageRank. That development led to the *TrustRank* & other techniques for preventing such attacks. PageRank had many variations that happened from time to time, as the developers realized the issues and problems with web pages. Though the part we know is just a basic idea and not the exact implementation, it still seems to work much more efficiently as compared to others.

II. Origin of PageRank

The project that led to a successful deployment of the PageRank Algorithm was started in 1995 by Larry Page & Sergey Brin while they were studying at Stanford. A similar algorithm that existed was Hypertext Induced Topic Search (HITS). It viewed pages in terms of "hubs" and "authorities" [1]. Another approach similar to PageRank dates back to 1941 when Wassily Leontief of Harward tried to divide the economy of the country into different sectors and ranked them on the basis of several parameters.

Fig. 1: A History of PageRank [1].

A. Traditional Information Retrieval

Long back, we used some totally different methods to retrieve useful information. Such methods existed before the birth of, what we call, *World Wide Web*. Some examples include searching for a book in the library catalog or searching for a particular topic in a professor's slides. These data collections were static, not inter-linked and much organized than the data we have today. A little later, we entered an era when data was being stored electronically, in CDs and on webpages.

But talking of today, the data storage methods have been computerized. Such computerized methods are known as *"Web search engines"*, virtual machines created by software and used to sort through virtual file folders to find relevant documents. The traditional information retrieval methods used 3 basic computer-aided techniques for searching:

1) Boolean Search Engine: It uses Boolean operators, AND, OR, and NOT, to combine logical statements. This works by taking into consideration some keywords and then checking the document for the presence or absence of those keywords. But this led to a problem. The document is considered either relevant or irrelevant. There is nothing like partial matching, which leads to the bad performance of such models.

2) Vector Space Model Search Engine: It stores files and data in the form of numeric vectors and matrices. Then various *Matrix Analysis Techniques* are used to find key features in the data. The vector space model allows documents to partially match a query by assigning each document a number between 0 and 1 [2]. As shown in Figure 2, the data is retrieved on the basis of a similarity index, which is calculated mathematically.

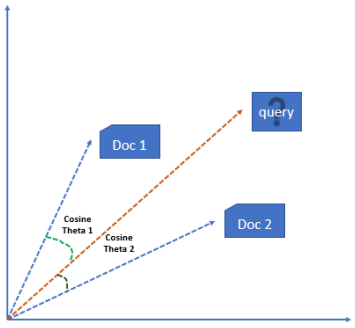

Fig. 2: Vector Space Model [5].

The *Generalized vector space model (GVSM)* is independent as compared to a normal vector space model. In other words, the involved set of vectors is linearly independent [8]. In this, it is not necessary that every pair of the vector is orthogonal to each other, but it is formed from smaller components of a particular collection [8].

3) Probabilistic Model Search Engine: It works by finding the probability by which a particular document is relevant for a user. It finds the ratio of probabilities of a document being relevant to a query to that of it not being relevant to that query, and this calculation is done recursively, to reach a final ranking. These models are complex and are thus not preferred, especially when the data terms are not independent.

A fundamental example of such a model is the **Bayesian model**. It answers the queries by finding out the "term frequency" of every distinct word in the document [10]. If d is the vector of *term frequencies* of words and a constant r depicts relevance of the word to a query, such that, $r = 0$ if the word is completely irrelevant and 1 if relevant, then, using the *Bayes' Rule* we can deduce the following result:

$$p(r|d) = \frac{p(r).p(d|r)}{p(d)}$$

where, $p(i)$ represents the probability of the word i. This way this model was used to calculate a score value and thus rank the web pages.

B. Web Search

Any information retrieval from the web is not a simple task, it involves many intricacies and subtasks, which eventually function in a particular fashion and a specific order to fetch the required results. Fig. 3 rightly describes the internal working of a search engine, starting from a collection of data to sending the required information to the end-user.

All web search engines have a *web crawler* which contains a software to collect data and categorize it using virtual robots or *spiders*. The crawler stores them in the *page repository* till they are sent to the *indexing module*. The more often required pages are stored in the repository for longer. The task of the indexing module is to extract useful information from the uncompressed pages and store them as "Cliffnotes", i.e. in a compressed form. *Indexes* hold the vital information for each of those pages [2].

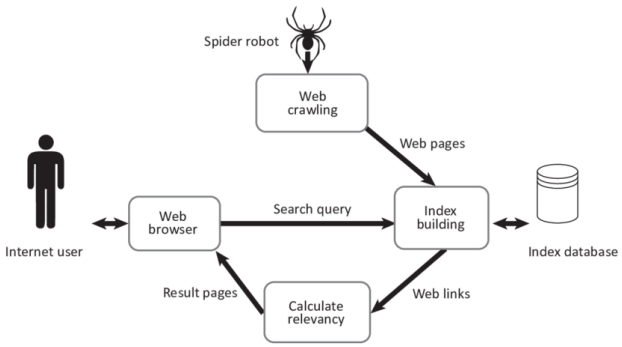

Fig. 3: Search Engine [6].

There are 3 types of indexes:

- Content Index
- Structure Index
- Special Purpose Index

Another tool, the *Query Module* converts the user's query to a different form that can be understood by the system and then consults various indexes to answer the query. *Ranking Module* ranks the available options or pages according to their relevance, by combining 2 scores, the **Content score** and the **Popularity score**, into an **Overall score** of a web page.

III. PageRank

Researchers Page & Brin realized the potential of a graph and how it can transform the existing search algorithms, and brought something called "PageRank". PageRank can be simply defined as a function that assigns real numbers to every page on the web. The crux is that higher the PageRank, more "important" the page is. There is no fixed method to calculate this score, many variations exist. And a small alteration can bring a visible change in the order of the pages.

A. Google Toolbar

Google toolbar[1] shows an approximation of ranks using their PageRank scores. The toolbar, once downloaded, resides in the browser, and shows the bar graph, depicting the score out of 10, Google home page has been assigned a score of 10/10, and others are given somewhere in between. It automatically updates the score whenever we switch from one page to another. This involves sending the information of each page visited the Google servers, thus, it seemed a violation of privacy policy. But, Google's privacy policy states that it does not collect any personal information like your name or email address. And the toolbar also offers a feature to disable the PageRank calculator, keeping the other functions as the were.

IV. Implementation of PageRank

In earlier days, graph was considered as an unexploited source of information. Most of the people could not think how web search results can be related to the graph. But think of the Web as a directed graph, where pages are the nodes, and there is an edge from page p1 to page p2.

As shown in Fig. 4, page 1 has links to page 3, page 2 and page 4, similarly, page 3 has a link to page 2, and so on.

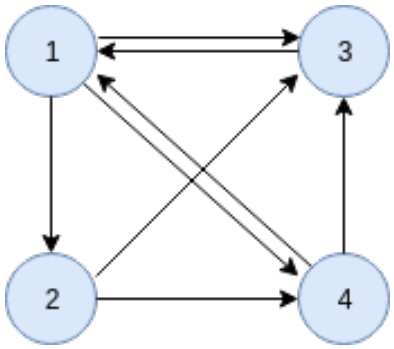

Fig. 4: A hypothetical example of the web

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

A. Mathematics of PageRank

We assume a *random surfer* starts at page 1 in Fig. 4. There are links to pages 3, 4 and 2, so the surfer can go to these pages with a probability of 1/3 and the probability of being on page 1 is zero. To generalize, we define a *transition matrix of the Web* to describe what happens to the random surfers after each step. If there are *n* pages, the graph has *n* nodes, and thus, *n* rows and *n* columns in the matrix. An element m_{ij} in i^{th} row and j^{th} column has a value $1/k$ if the page j has k out links, and one of them is to page i, else $m_{ij} = 0$. The transition matrix for the above graph would look something like this:

$$\begin{bmatrix} 0 & 0 & 1 & 1/2 \\ 1/3 & 0 & 0 & 0 \\ 1/3 & 1/2 & 0 & 1/2 \\ 1/3 & 1/2 & 0 & 0 \end{bmatrix}$$

Here, the first column represents the weights of edges (probabilities) coming out from node 1, and so on. The probability distribution of the random surfer's position is described by a column vector, whose j^{th} element is the probability that the surfer is at page j. Thus, probability is the idealised *PageRank* function [7].

To begin, we initially assign equal probabilities to all n pages, i.e., $1/n$. Thus, the initial vector v_o has all components equal to $1/n$. Let the transition Matrix be M. After one iteration or one step of the surfer, the vector will be Mv_o, and after 2 steps it will be M^2v_o.

This holds true because we know that if a page has in links from k pages, it PageRank at every iteration is calculated as,

$$PageRank_{k+1}(P_i) = \sum_{j=0}^{k} \frac{PageRank_k(P_j)}{|P_j|}$$

Where k is the iteration no. and $|P_j|$ denotes the no. of out links from page j.

Now, using the theory of *Markov Processes*, we calculate the probability that the surfer will be at node i at the next step as follows:

$$P(i) = \sum_j m_{ij} v_j$$

This method gives the same result as the previous equation, just in a simplified form. Now let's use this method to the graph in Fig. 4. There are 4 nodes. Therefore, initially, the vector v_o has all values as 1/4. And then at each step, we multiply v_o by the matrix M. The results are as follows:

$$\begin{bmatrix} 1/4 \\ 1/4 \\ 1/4 \\ 1/4 \end{bmatrix} \begin{bmatrix} 9/24 \\ 5/24 \\ 5/24 \\ 5/24 \end{bmatrix} \begin{bmatrix} 15/48 \\ 11/48 \\ 11/48 \\ 11/48 \end{bmatrix} \begin{bmatrix} 11/32 \\ 7/32 \\ 7/32 \\ 7/32 \end{bmatrix} \cdots \begin{bmatrix} 3/9 \\ 2/9 \\ 2/9 \\ 2/9 \end{bmatrix}$$

Here, the values of probabilities or the PageRank scores do not vary much, but in reality, when we have millions of pages to deal with, these values are relatively different and can be ranked easily according to these values. This was one of the initial attempts to find the PageRank, many variations of this algorithm exist.

B. Problems associated with the Iterative method

- Cycles: If the graph has a cycle, say there are 2 nodes, pointing to each other, then the transition matrix will keep rolling between [0 1] and [1 0]. Thus, the cyclic graph won't converge.
- Dangling nodes: Nodes that do not have any outgoing link, will not get an accurate score using this method because the entire column corresponding to that page will be zero.

C. Early Modifications to the Method

Brin and Page improvised this initial method to increase accuracy and solve the issues faced. They came up with a new matrix, G, which they called *Google Matrix*. It is defined as follows:

$$G = \alpha M v_o + \frac{(1-\alpha)}{n}$$

Where α is a scalar between 0 and 1, known as the *damping factor*, and n is the total number of pages. This method solves the problem of dangling nodes as when there are zero outgoing links from a node, or in other words when the random surfer is stuck on a page from where he has no path to go to some other page, he jumps to a random page and starts walking.

Mathematically, when the term $\alpha M v_o$ becomes zero, G stays non-zero due to the term $(1-\alpha)/n$, and thus the matrix does not die out and the iterations can continue.

Similarly, when there are cycles, the extra term will ensure that on each iteration, the matrices do not keep switching between 0's and 1's, forever and reach a distinct positive value.

D. The Damping Factor, α

The developers, Page and Brin suggest setting the value of α as 0.85. The answer to why this value, and not any other value of α, is that as $\alpha \to 1$, the expected number of iterations required by the power method increases [2]. The tolerance becomes too small if α is small. Thus there is a trade-off between efficiency and effectiveness, balancing which the duo came up with the value 0.85 for α.

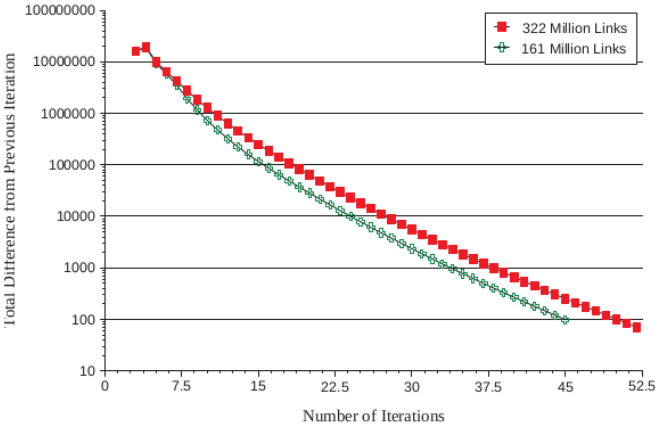

Fig. 5: Convergence of PageRank Computation [3].

V. Applications of PageRank

A. Searching

The most common application of the PageRank Algorithm is searching. It was with this purpose that Brin and Page designed it for Google. Whenever we have a query or want to know about something, Google is what comes first to our mind. But how does the search work? Google has hundreds of pages lined up related to your query, how does it decide which one should be kept at the top and which one later on while displaying the results? All this is managed by algorithms such as PageRank, though some sources say that Google does not use PageRank anymore, but the basic idea used for searching still has its roots in PageRank.

It is assumed that Google uses over 250 different properties of pages, from which a linear order of pages is decided whenever a user makes a query [7]. One of the properties being the words in the query. The result pages should have some words in common. The PageRank score is calculated for every individual web page and they are displayed in that sequence. The other properties which are taken into account while calculating this score are the presence of such keywords in the main headings and headers of the page.

B. Web Advertising

We all have encountered numerous ads on pages, which sometimes even become annoying when they hinder user experience in any form. There is also confusion when it comes to deciding which ads need to be displayed on the page in terms of relevance, cost, previous searches, etc. Figure. 6 rightly describes how PageRank is helpful in web advertising. Every web page follows certain rules and protocols for testing before displaying any ad.

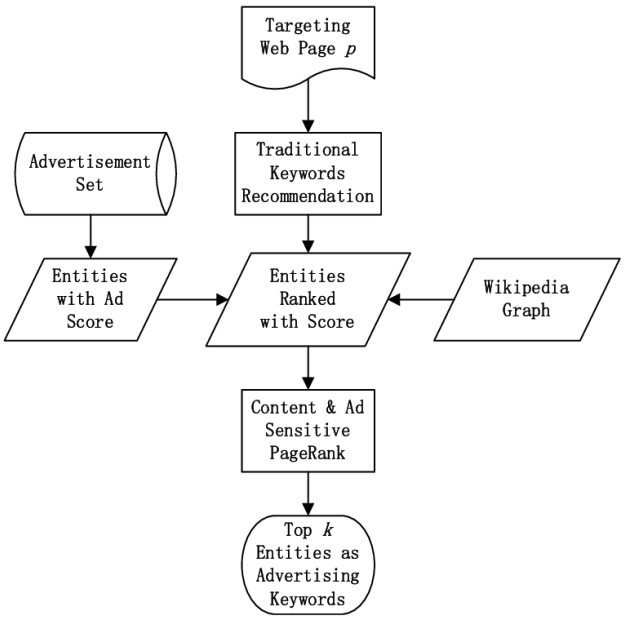

Fig. 6: Web advertising testing [14].

Mainly websites consider the following points in deciding this:

- Earning on each click on an ad.
- No. of clicks on that particular ad per minute.
- Search history of a particular user.
- Budget of each ad display.

Most web sites use the *Greedy Algorithm* approach to solve this problem, as there are many trade-offs while prioritizing any of the above points. In some cases, ads need to pass the PageRank test for *Search Engine Optimization*, through which the page decides to show that particular ad or not [12].

VI. Issues in Large-Scale Implementation of PageRank

1) **Storage Issue**: PageRank requires a lot of memory space to store information at each level. The information includes web pages, their locations, hyperlink graphs, image indexes, content score information, and PageRank scores. All this information becomes huge as the size of data increases.
2) **Convergence**: It is never completely certain where the values of the PageRank score will converge. But as the developers said, the values do not signify anything. What matters the most to us, is the sequence, i.e., the order in which the score decreases, as this order finally determines the rank of that page.
3) **Accuracy**: There's nothing right or wrong in the PageRank applications. Taking the example of Google, we have no means to say that the results shown by Google are ranked in an incorrect manner or are completely accurate. The value of the parameter α has been decided to be 0.85 to be the most accurate when talking of large computations.
4) **Dangling Nodes**: We have today, many ways to handle dangling nodes, one of which we have discussed in this paper previously. The presence of these dangling nodes causes both philosophical and computational issues for the PageRank problem.
5) **Recently Developed Pages**: Whenever a new page is introduced, there are very few hyperlinks pointing to it (possibly zero). This leads to a situation of a zero row in the transition matrix. Thus, such a row on multiplying with the non zero column vector would give a very small value. This will lead to a lesser PageRank score for that page always, thus Google uses some separate variation to handle such pages.

VII. The HITS Method

HITS was a method used to rank web pages before PageRank was born. It is said to have the closest connections to PageRank, in terms of implementation. HITS refers to *Hypertext Induced Topic Search*. It was introduced by Jon Kleinberg in 1998 when Page and Brin were working on the PageRank algorithm. HITS also makes use of the hyperlinks to calculate the page popularity.

There are a few differences between PageRank and the HITS Algorithm:

- PageRank gives one final score, whereas HITS method gives two.
- HITS depends very much on the user's query unlike PageRank.

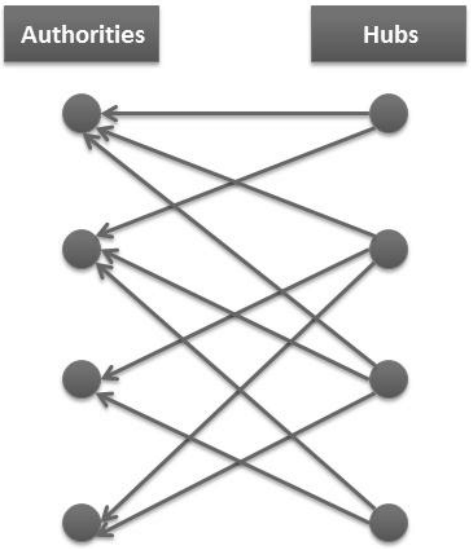

Fig. 7: HITS authorities and hubs [13].

HITS thinks of pages as authorities and hubs. Here, authority refers to a page which has many in-links and hubs are pages with many out-links.

HITS calls a page *good* if it follows the following criterion:

Good authorities are pointed to by good hubs and good hubs point to good authorities [2].

Thus, every page has an authority score as well as a hub score. Let the authority score of page i be denoted by x_i and its hub score be y_i. E is the set of edges of in the directed web graph where e_{ij} denotes a directed edge from page i to page j. Initialise: $x_i^{(0)}$ and $y_i^{(0)}$. The equations used to compute their values further as follows:

$$x_i^{(k)} = \sum_{j:e_{ji}\epsilon E} y_j^{(k-1)} \; and \; y_i^{(k)} = \sum_{j:e_{ji}\epsilon E} x_j^k \; for \; k\epsilon N$$

These are *Kleinbergs original equations* [2].

VIII. Future of PageRank

After so many years, PageRank is still in use. From the day Google unleashed its famous algorithm, we have been trying to modify it as per requirements and it has, in most cases, proven to work great. PageRank can have even greater no. of applications and can be studied in depth if we are able to visualize its working at each node. We discuss this visualization aspect in detail in a separate subsection.

A. Visualization of of PageRank

There have been many attempts, in the past as well as now, to visualize the PageRank scores. One of them being the famous Google Toolbar, Figure. 7. But this was a case in which making the ranks visible created a problem.

Also, there is even more to visualizing the algorithm, than just showing the final score values. The user might also want to know the hyperlink graph, the score of all the connected pages (nodes) and the modification in this score each time a new node is added or deleted. If we are to implement our own visualization tool for an algorithm like PageRank, we need to keep a few important points in our mind:

1) How adding new links will change the score for existing nodes?
2) How will deleting a connected node affect the other nodes?
3) Changing the incoming and outgoing links of an already existing page.

IX. Conclusion

PageRank is much more than simply a page ranking algorithm. It has a wide range of applications in terms of searching, ranging from a particular query, to web advertising. Policies and protocols for the handling, storage and application of big data are going to need to catch up with the analysis capabilities that already exist. The basic implementation of PageRank can be done in any language and framework, though experimentally it can be seen that using broadcasting in matrix multiplication makes it 10 times more efficient than multiplying it term by term. When done using the Hadoop MapReduce framework, the efficiency improves further, and even further if we use Spark. There still exists a huge scope of improvement.

References

[1] A History Lesson on PageRank, https://bit.ly/2AOVDk6
[2] Amy N. Langville, and Carl D. Meyer, "Google's PageRank and Beyond", Ch. 1, Section 1.2.2
[3] Sergey Brin, and Larry Page, 'The PageRank Citation Ranking: Bringing Order to the Web', January 29, 1998
[4] Basic Search Tips and Techniques: Gale Databases, https://libanswers.an.edu/ANU/faq/103964
[5] Information retrieval document search using vector space model in R, https://bit.ly/2MeXt38
[6] Competitive Analysis of Retail Websites through Search Engine Marketing, https://bit.ly/2OuCU5r
[7] Jure Leskovec, Anand Rajaraman, and Jeffrey D. Ullman, "Mining of Massive Datasets"
[8] Hai Dong, Farookh Khadeer Hussain, and Elizabeth Chang, "A Survey in Traditional Information Retrieval Models"
[9] Konstantin Avrachenkov, and Nelly Litvak, "The Effect of New Links on Google Pagerank, Stochastic Models", September 2006, https://doi.org/10.1080/15326340600649052
[10] Ernest P. Chan, Santiago Garcia, and Salim Roukos, "Probabilistic Modeling for Information Retrieval with Unsupervised Training Data", 1998, AAAI
[11] Google PageRank, https://www.seocu.org/pagerank/
[12] "Do ads pass PageRank for Search Engine Optimization", https://bit.ly/33qAzg9
[13] HITS Algorithm, https://bit.ly/2MysMpW
[14] Advertising Keywords Recommendation for Short-Text Web Pages Using Wikipedia, https://bit.ly/319cqcj