# How Google knows what you are looking for

Hildebrando M. Rodrigues[1], Guilherme M. Alves[2], Matheus C. Nali[2], Victor T. B. Shime[2]

[1] Institute of Mathematical and Computational Sciences
[2] São Carlos School of Engineering
University of São Paulo
hmr@icmc.usp.br, guilherme.malacrida.alves@usp.br, matheus.nali@usp.br, victor.shime@usp.br

**Abstract**

*This article is part of a scientific initiation program, under the supervision of professor Hildebrando Munhoz Rodrigues, in which students Guilherme Malacrida Alves, Matheus Carvalho Nali and Victor Takao Bernadino Shime studied the functioning of search tools such as Google and its algorithms with highlighting the PageRank where the mathematical fundamentals used in its development are explained and illustrated.*

## 1. Introduction


**Figure 1:** *Search engine*

First, we inform that the results here are not new and the purpose of this article is informative, showing the importance of linear algebra, matrix theory and spectral analysis in the development of the Google search system.

Now, let's take a brief history of Google [1]. Sergey Brin and Larry Page (Google Creators) met at Stanford University while studying for a PhD in computer science. At that time, Page had a crazy project to download the entire Internet on his computer in just over a week. But after a year he had completed only a portion of it. Well, we could say that Page was very optimistic, couldn't we?

It turns out that in this project Sergey joined Page because he was interested in mining the collected data and that's when they started working hard to develop Google. In a short time, their search system was used about 10,000 times a day at Stanford. That way, they realized they needed more computers to support the system, and so they decided to build the company in Silicon Valley.

But even using low-cost hardware, there wasn't enough money to buy the equipment. To help them, one of their teacher (David Cheriton) contacted Andy Bechtolsheim (Investor and co-founder of Sun Microsystems) recommending to met Sergey and Larry.

So, Andy met their project and became very interested believing in its potential to surpass the search engines from that time (for example the AltaVista). In addition, he admired the choice of Sergey and Larry to invest the money in low-cost components to build the computers rather than spend on marketing or more expensive equipment. That way, in 1998 Andy made a check of $ 100,000 becoming the first Google investor.

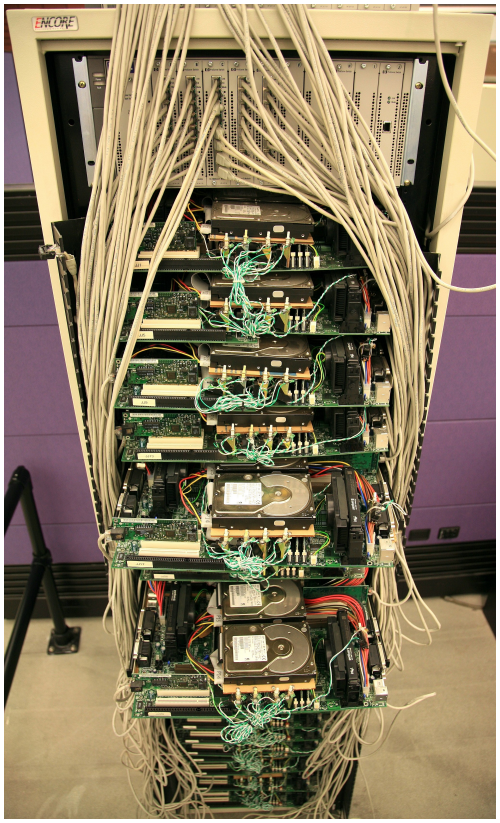


**Figure 2:** *First Google Production Server - Constructed with low cost hardware.*

Have you noticed that, in general, search engines have what you are looking for in the first results? or that a virtual encyclopedia is always at the top of your search results? This is not any kind of black magic, it's simply mathematics (more specifically linear algebra) and computing.

Now, how does this work? One of the factors analyzed in this article is the importance ranking of each web page related to its search using a method called "PageRank", which basically assigns a value to each page representing how relevant it is between all pages found. We know that the internet have a huge amount of data. Because of that, to be possible to analyze all information efficiently and effectively, it was necessary to find a method that would decide which pages are most important without a person having to read all of them and sort, something that, in addition to being inefficient, depends heavily on the personal opinions of the person doing the classification.

The solution was to make the pages decide which one is the most important through the

links between them, which are the way the pages relate. The "PageRank" algorithm uses the logic that the importance of a page is defined by the pages that link to it. A more detailed description of this method can be found in [2].

Subsequently, several modifications were made in the "PageRank" algorithm to increase its efficiency, as proposed by Golub [5]. He noticed that the importance rank value was determined faster for the less important pages while the more important ones took longer. With this he proposed to stop calculating the less important pages at some point since their value was close enough to the real value. In this way it was possible to increase the speed of the algorithm by almost 30 %. For simplicity we will not use this modification here.

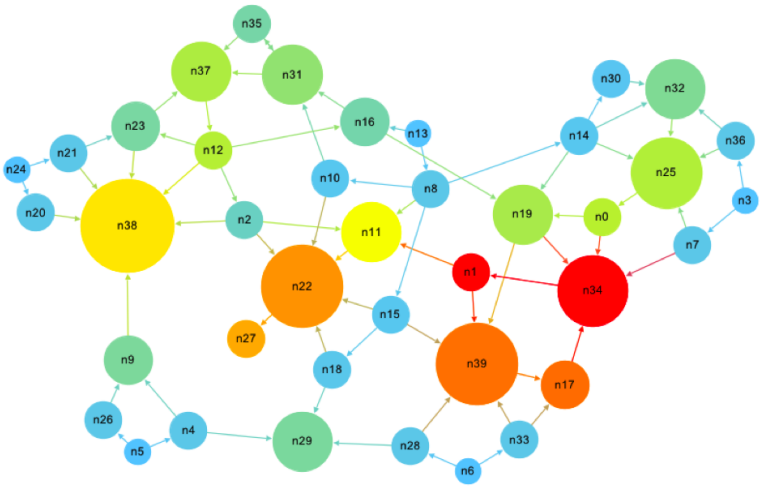


**Figure 3:** *Graph representing the links between webpages*

## 2. Page Rank Method

To illustrate the Page Rank Method, we will start with an example of a four-page network (1,2,3,4) whose links are described in the image below. The importance of each page will be denoted by $x_1, x_2, x_3, x_4$ respectively.

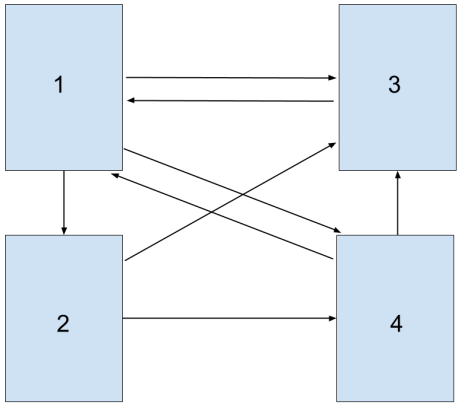


**Figure 4:** *Network of pages with arrows representing links*

A simple way to interpret the algorithm is to say that a page has a sum of votes equal to its importance and divides those votes equally between the pages to which it has links. The importance of a page is then the sum of all the votes it receives. Summarizing in one equation, where $n_j$ is the number of links in the page $x_j$, we have:

$$x_i = \sum_{j \neq i} \frac{1}{n_j} x_j$$

For the case of the previous figure we can write the importances as being:

$$\begin{cases} x_1 = x_3 + \frac{1}{2}x_4 \\ x_2 = \frac{1}{3}x_1 \\ x_3 = \frac{1}{3}x_1 + \frac{1}{2}x_2 + \frac{1}{2}x_4 \\ x_4 = \frac{1}{3}x_1 + \frac{1}{2}x_2 \end{cases}$$

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & \frac{1}{2} \\ \frac{1}{3} & 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{3} & \frac{1}{2} & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

We will use matrix notation to simplify:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 1 & \frac{1}{2} \\ \frac{1}{3} & 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{3} & \frac{1}{2} & 0 & 0 \end{bmatrix}$$

This system is a type of problem called the eigenvalue problem. Eigenvalue problems are equations of type $Mx = \lambda x$ where M is a matrix and $\lambda$ is a constant, and it can have a single solution where all components of x are zero or infinite solutions, which depends on M and $\lambda$. If

we have infinite solutions we say that $\lambda$ is an eigenvalue of M and the nonzero values of x that satisfy are eigenvectors.

Knowing this, in order to the system have a solution, we need to ensure that 1 is eigenvalue of the **A** matrix that we constructed with the links in our network, let's use the result below and the Perron-Frobenius theorem.

THEOREM: A matrix A whose entries are all nonnegative and the columns sum 1 (coincidentally, or not, as we have constructed) has 1 as eigenvalue. As is shown in [2].

With this theorem we know that our problem has a solution that is not zero. But we still have to make a unique solution for it because computers can't calculate infinite solutions and we do not want to have different results every time we do a search. The Peron-Frobenius theorem gives us a solution to this, but first let's look at some useful definitions.

## 3. Useful Definitions

- Spectral radius: For this specific case, assume that it is the largest module among the modules of the matrix eigenvectors, which is a simplified way of defining the spectral radius.
- Non-negative Matrix: A matrix with all elements greater than or equal to zero.
- Positive Matrix: A matrix with all elements greater than zero.

## 4. Perron-Frobenius Theorem

PERRON-FROBENIUS THEOREM: Let **M** be a non-negative square matrix, the spectral radius of M, $\rho(\mathbf{M})$, is the largest eigenvalue in module and has no eigenvector negative. In addition, no other eigenvalue of **M** is greater in absolute value than $\rho(\mathbf{M})$ nor has positive eigenvector. If **M** is positive we will have that the eigenvalue is simple. [4]

Now let's go to the explanations of the theorem. What is the use of this theorem? First, It shows that there is a single eigenvalue with positive eigenvector, so all positive eigenvectors are multiple of each other. Because of that, we just choose any one of them with the sum of their inputs equal to 1 for example (it is important to have a well-defined criterion to avoid ambiguities), and we have a unique solution to be found. Having a single solution is essential in computational applications so they are viable on a large scale with fast processing. Without it it would not be possible to handle all the pages of the internet and even if for a simple search it would be very difficult to find without knowing the address of the page.

Second, some problems can arise in the calculation of this eigenvector, such as pages that do not receive any links (dangling nodes) and groups of pages that have no links to each other. These situations generate problems with the uniqueness of the solution that can be solved by making some changes to the matrix **A** (These cases are more detailed in [2]). In the case of unrelated page groups the solution is to create a new matrix from **A** which have all the positive entries, but still has a sum of 1 in the columns, so that the second part of the theorem guarantees that the solution is unique. The case of pages that do not receive links is more problematic to be handled if you did not want to simply delete them from your list.

Last, these results are not as immediate as written here, or Google's creators would not have gotten rich with it. Exemplifying the importance of this is the fact that the PageRank of 26 million pages can be calculated in a few hours by a medium workstation as quoted by the Google creators themselves in [3], considering the computational power of the time when the result was published.

## 5. Back to example

Now, to solve the eigenvalue problem for the given example, we compute the eigenvectors of the matrix **A** corresponding to the eigenvalue 1. Returning to the constructed system and leaving the answer in function of $x_1$.

$$\begin{cases} x_1 = x_3 + \frac{1}{2}x_4 \\ x_2 = \frac{1}{3}x_1 \\ x_3 = \frac{1}{3}x_1 + \frac{1}{2}x_2 + \frac{1}{2}x_4 \\ x_4 = \frac{1}{3}x_1 + \frac{1}{2}x_2 \end{cases}$$

The first value is obtained immediately from $x_2 = \frac{1}{3}x_1$, next we have:

$$x_4 = \frac{1}{3}x_1 + \frac{1}{2}x_2 = \frac{1}{3}x_1 + \frac{1}{6}x_1 = \frac{1}{2}x_1$$

And finally:

$$x_3 = \frac{1}{3}x_1 + \frac{1}{2}x_2 + \frac{1}{2}x_4 = \frac{1}{3}x_1 + \frac{1}{6}x_1 + \frac{1}{4}x_1 = \frac{3}{4}x_1$$

Which results in:

$$\implies \begin{cases} x_1 = x_1 \\ x_2 = \frac{1}{3}x_1 \\ x_3 = \frac{3}{4}x_1 \\ x_4 = \frac{1}{2}x_1 \end{cases}$$

If we conveniently use $x_1 = 12$, we get: [12 4 9 6]. Normalizing these values so that the sum is 1, we obtain the new eigenvector (remembering that any nonzero multiple of an eigenvector is also an eigenvector).

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \approx \begin{bmatrix} 0.39 \\ 0.13 \\ 0.29 \\ 0.19 \end{bmatrix}$$

With this, the most important page is 1! But why not 3? page 3 receives links from all 3 other pages as we can see from the image. What happens is that although page 3 receives links from others, it links only to page 1 which impacts on its result of importance. It is important to remember that the importance is associated with the relationship between the pages, that is, with the links that come and arrive in it, and in this case, the whole importance of page 3 is passed to page 1, which is the only one that receives link of 3. Thus, 1 is the most important, since it must have at least the same importance of 3 as the calculations prove.

With this, we conclude that using theorems, convenient models and necessary adaptations for certain cases it is possible to analyze and classify the pages of a search. In addition, it is fundamental to optimize the algorithms to improve the use of computational resources since we are talking about a huge amount of analyzed pages.

Finally, it is important to understand that PageRank sorts pages according to their **importance** and many other algorithms are also used in the search process in order to try to present the result you want on the first page.

Other important algorithms in page ranking:

- Hilltop Algorithm
- Topic-Sensitive PageRank

## References

[1] Vise, D. A., Inside the Hottest Bussiness, Media and Technology Sucess of Our Time, PAN, 2005.

[2] Bryan, K., & Leise, T., The $25,000,000,000 Eigenvector:The Linear Algebra Behind Google, Society for Industrial and Applied Mathematics, 2006.

[3] BRIN, S., & PAGE, L.,The anatomy of a large-scale Hypertextual search Engine. http://www-db.stanford.edu/ backrub/google.html

[4] KATO, T., A short introduction to linear perturbation theory, Springer-Verlag, 1982.

[5] KAMVAR, S., HAVELIWALA, T., GOLUB, G., Adaptive methods for the computation of PageRank, Linear Algebra and its Applications 386 (2004) 51–65