Ukrainian Catholic University

Applied Sciences Faculty

Data Science Master Programme

THE $25,000,000,000 EIGENVECTOR

Project report

Authors:
ROMAN RIAZANTSEV
NAZARIY PEREPICHKA

February 18, 2018

Abstract

Google PageRank algorithm was a key element of the Google success in the early stages of the company's development. Since the core of the process related to linear algebra, we can go through the algorithm and understand its beauty, simplicity and prerequisites for a breakthrough in the search engine market.

1 Introduction

This project provides a brief overview of principles behind PageRank. Understanding of those concepts can help readers to promote their website since the higher you page gets in Google rank, the more people will visit your site. Due to pure mathematics core of the topic, such work can help to build an intuition for different Linear Algebra topics.

Everyday usage of Google makes it easy to forget how fascinating their search engine is. Query transforms in relevant results in a second. Behind this operation lies a beautiful mathematical idea.

Before Google came in the game with relatively sophisticated search engine architecture, the web was filled up with algorithms that merely counted the number of words matching the query. Even though the simplicity of such approach could bring benefits regarding computational speed, the search results were not accurate, and it is easy to understand why. For example the top result for query "How to find matrix inverse" would be the page with phrase "How to find matrix inverse" repeated maximum amount of times. No surprise that with such search systems user would need to go through the countless sequence of results until she rich desired page. The situation changed due to PageRank algorithm invented by Larry Page and Sergey Brin.

Let's shed some light on the algorithm, which changed the way people search for information.

2 LA behind PageRank

2.1 PageRank Overall

In this article, the term "score" will describe the measure of page importance, relative to other nodes in the graph. It is crucial to mention that value of such parameter cannot be less or equal to zero. The essence of PageRank

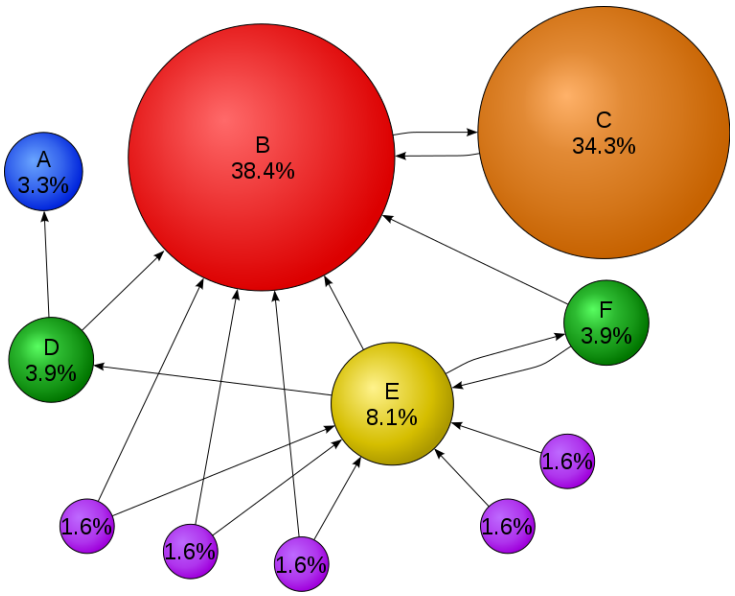

Figure 1: Mathematical PageRanks for a simple network, expressed as percentages. (Google uses a logarithmic scale.) Page C has a higher PageRank than Page E, even though there are fewer links to C; [2].

is: every page equally shares own importance between linked sites, and every page gain importance by accumulating shared scores. In such way, the internet becomes a system in which pages can vote for each other and "choose" direction of the score distribution. Every page has set of links to it - sources of a gained score, such links called the backlinks.

Let the number of sites on the internet equal to n, and each page in that web will be indexed by a number k, $k \in [1, n]$. Simplified example is illustrated on Figure 1: every row represent a link from page to page. Let x_k be the notation for the importance score of the site with index k. Page j called more important than page k if $x_j > x_k$.

The easiest way to calculate x_k which came to mind is to do so based on the number of backlinks for page k. Thus from looking at Figure 1 we can see that that page 3 has the principal amount of backlinks and conclude that it is the most important page. Every backlink of k can be considerate as a vote for the higher score from linking page. But it is evident that such approach completely ignores the importance of a voting page. For example, a backlink from "Wiki" will share more importance with page than link from

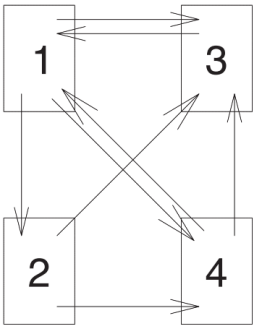

Figure 2: An example of a web with only four pages. An arrow from page A to page B indicates a link from page A to page B.[1]

some no-name website

Therefore the right way of calculating x_k is to take into account weight of linking page. To do so, we can use following equation:

$$x_k = \sum_{j \in L_k} \frac{x_j}{n_j} \tag{1}$$

where n_j is the number of outgoing links from page j (which must be positive since if $j \in L_k$ then page j links to at least page k!). We will assume that a link from a page to itself will not be counted.

To take every aspect of this in a count let's create matrix $n \times n$ matrix A, where a_i represent information about links of the page with index i, $a_i j = 1/n$, where n - the number of pages for which i votes. That is a_j provide information about backlinks of page j. $A_i j$ is an amount of score which page i share through the link with page j. Now we have the matrix which contains all desired information. Important to mention that A is an example of the column-stochastic matrix, which means that all entries of ai sums to 1. Next step is to find the eigenvector of A. In our case $x = Ax$. Then to normalize;

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

We also can look at situation from dynamical systems point of view:

Let v be an initial vector with where score is equally distributed among n pages. In our case with 4 pages each page will have score of 0.25. After each multiplication of A by v, scores reassigned to v based on how many importance was added to that pages by its backlinks[6]. Step one can be notated as $v_1 = Av$. Step two notation is: $v_2 = A(Av) = A^2v$. Numeric computations give:

$$v = \begin{bmatrix} 0.25 \\ 0.25 \\ 0.25 \\ 0.25 \end{bmatrix}, Av = \begin{bmatrix} 0.37 \\ 0.08 \\ 0.33 \\ 0.20 \end{bmatrix}, A^2v = A(Av) = A \begin{bmatrix} 0.37 \\ 0.08 \\ 0.33 \\ 0.20 \end{bmatrix} = \begin{bmatrix} 0.43 \\ 0.12 \\ 0.27 \\ 0.16 \end{bmatrix}, \tag{3}$$

$$A^3v = \begin{bmatrix} 0.35 \\ 0.14 \\ 0.29 \\ 0.20 \end{bmatrix}, A^4v = \begin{bmatrix} 0.39 \\ 0.11 \\ 0.29 \\ 0.19 \end{bmatrix}, A^5v = \begin{bmatrix} 0.39 \\ 0.13 \\ 0.28 \\ 0.19 \end{bmatrix}, \tag{4}$$

$$A^6v = \begin{bmatrix} 0.38 \\ 0.13 \\ 0.29 \\ 0.19 \end{bmatrix}, A^7v = \begin{bmatrix} 0.38 \\ 0.12 \\ 0.29 \\ 0.19 \end{bmatrix}, A^8v = \begin{bmatrix} 0.38 \\ 0.12 \\ 0.29 \\ 0.19 \end{bmatrix} \tag{5}$$

The vector v which not affected anymore by multiplication is PageRank vector.[6]

2.2 Problems with PageRank computation

Even though for our graph on Figure 2 with 4 nodes such algorithm clearly succeed there is 2 cases in which problems arise and calculation of PageRank require additional steps.

In first scenario our web consist of two isolated subwebs and not a single page from subweb 1 links to subweb 2 and vice versa. Such situation leads to vote-matrix with where $V(A) > 1$, which is lead to contradiction, since that matrix will have at least 2 eigenvectors. Which of them we must use to ranking pages? Maybe we must use their linear combination?

In the case when web contains dangling nodes (nodes which have no links to other pages) matrix A will have one or more columns with all zeros. Matrix A will have all eigenvalues less or equal to 1 in magnitude. But 1 not

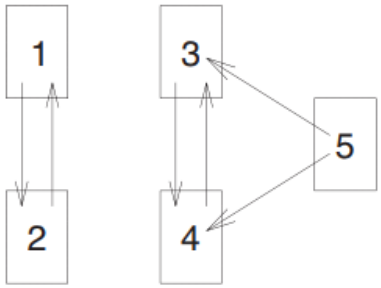

Figure 3: A web of five pages, consisting of two disconnected "subwebs" W1 (pages 1 and 2) and W2 (pages 3, 4, 5).[1]

always will be the eigenvalue of such matrix. The PageRank of the net with dangling nodes still can be calculated by similar methods to those which have been discussed. The corresponding substochastic matrix must have a positive eigenvalue $\lambda \leq 1$ and a corresponding eigenvector x with non-negative entries (called the Perron eigenvector) that can be used to rank the web pages.[1]

2.3 Solutions of subwebs problem

For any web which consists of n nodes, none of which is dangling we can calculate matrix M:

$$M = (1 - m)A + mS \tag{6}$$

Such representation is convenient in case of subwebs. S is an n x n matrix where all entries equal to 1/n. Apparently, the dimension of S equal to 1 and this matrix is column-stochastic. Parameter m must satisfy $0 \leq m \leq 1$. The value of m originally used by Google is reportedly 0.15 [1]. Matrix M always will be one dimensional. Therefore we can use it to calculate scores for the web with multiple subwebs. The closer value of m to 1 the more equally values of M become distributed. It is clear that if m=1 M=S and all scores of PageRank are equal. Matrix M does not solve a problem of dangling nodes since matrix M still substochastic for any m¡1. The equation x = Mx can also be cast as

$$x = (1 - m)Ax + ms \tag{7}$$

where s is a column vector with all entries $\frac{1}{n}$. Note that $Sx = s$ if $\sum_i x_i = 1$.

2.4 Internet surfing is described by a Markov process

PageRank has the probabilistic part to it, since assigning importance scores can be considered as markov process. For example, there is a web of n pages, none of which is dangling nodes.[1] The surfer begins from the random initial page and then switch to other by a random link on this page. That mean surfer has equal chance to switch on any of the linked pages. To be concrete, if the page has r links, there is equally distributed probability equal to $\frac{1-m}{n}$ surfer will choose one of them and $\frac{m}{r}$ that surfer will choose random web page. PageRank value corresponding to the fraction of time which surfer spend on the page: more important page has high probability of visit and as consequence surfer more often visit such page.

3 Implementation of algorithm

The algorithm of PageRank computation is based on the power method. Important to mention that our implementation is allowing to use the different type of data structures for data storing. Because of the massive amount of zero entries in our stochastic matrix we can use so-called sparse matrix, but algorithm stays the same.

Steps:

1. Form initial matrix A
2. Rewrite it in stochastic form(M calculation)
3. Use power method to find PageRank vector
    - Form initial guess
    - Improve answer by multiplying current answer value on matrix,
    - Stop when distance between previous guess and current is small enough

Our implementation can be found at the github repo[3]. We've implemented it using Python. Our 34 datasets were taken from University of Toronto official site[4].

There are three implementations with the usage of three different data structures on our github repo. We're using simple nxn matrix, Coordinated Format(COO) Sparse matrix and Compressed Sparse Row(CSR) matrix.

We compare execution time of every part of an algorithm for every type of matrix, and we can see, that for our 34 datasets there is no principal difference in execution time, but sparse matrix allow us to reduce a lot of memory usage.

Detailed information about computation parameters can be found in output folder of our repo.

4 Conclusion

We discovered linear algebra, which stays behind the success of one of the biggest tech company in the world. We hope that we produced an article, which will help people to have a grasp on PageRank.

We strengthen our teamwork skills and were fascinated by the power of linear algebra. We implemented our solution flexible enough to use different data structures, and it allowed to reduce memory usage.

The main difficulty, which we faced, was finding appropriate data. At some point, we started to build our dataset based on the Ukrainian version of Wikipedia using the web crawler, but gladly we saw a lot of appropriate data on Toronto University website.

References

[1] Kurt Bryan and Tanya Leise: The $25,000,000,000 eigenvector The linear algebra behind Google,
https://www.rose-hulman.edu/~bryan/googleFinalVersionFixed.pdf

[2] Wiki: PageRank,
https://en.wikipedia.org/wiki/PageRank

[3] Nazariy Perepichka and Roman Riazantsev: Implementation,
https://github.com/baaraban/customPageRanchyk

[4] University of Toronto: Datasets,
http://www.cs.toronto.edu/~tsap/experiments/download/download.html

[5] Andrew Dynneson: Google's PageRank algorithm powered by linear algebra,
https://goo.gl/fmvSpK

[6] Raluca Remus: PageRank Algorithm - The Mathematics of Google Search,
https://goo.gl/Bch6pk