Google PageRank algorithm (Link analysis)

Parsa Tasbihgou
Prof. A.Jamshidi
University of Tehran, department of computer science
and mathematics

July, 2021

Abstract

The PageRank algorithm was the foundation of google company by Sergey Brin and Larry Page. This algorithm was a revolution in internet searching due to its speed and quality of results. However the original algorithm has changed a lot since its invention in 1998, the basics idea has been the same. In this text we will take look at the algorithm with focus on the underlying mathematics. Later following the idea of graph centrality

To better understand the discussions in this text a basic understanding of graph theory and matrices especially stochastic (probability) matrices will come in handy.

Introduction

With the continues growth of the internet, searching content becomes more difficult and more efficient algorithms are required. Before the PageRank algorithm was invented by Sergey Brin and Larry Page (who later on the basis of this algorithm founded google company), most search engines used keyword search algorithms and citation counters, which were slow and limited due to the need to scan large amounts of text and absence of clustering methods at the time, also the generated results were often not satisfactory.

The PageRank algorithm, however used the structure of the internet and the corresponding link graph to assign ranking to individual webpages so the results from a keyword search could be prioritized. The basic idea behind the PageRank algorithm is that the links between webpages act as votes that pages give to each other, so when a page has many link to it, it must have valuable content otherwise the content of the page is not very popular or useful. The ranking of the pages change the value of their vote so if a high rank page has a link to a page this link is more valuable than other links to that page; for example if a very popular and credible website like CNN has a link to a webpage A, page A gains a lot of points from this link since CNN has a high rank. Also webpages divide their votes between the pages they have links to, so if a page has many outgoing links each link has a lower value, and if a page has just a single outgoing link the receiver of that link is getting all the votes from that webpage.

For simplicity and generality purposes we will consider the internet a directed graph and the discussions will be in graph theoretical terms and the corresponding matrices.

To simulate the internet with a graph, each webpage is mapped to a node in the graph and a link from page A to page B is a directed edge from node A to node B, later on we will assign values to the vertices and the edges so a weighted directed graph is produced and we will use that graph to analyze the internet and generally any network system.

Before we get to the formal definition of ranking and the PageRank algorithm, let's see an intuitive explanation of the ranking method we want to define:

Let *G* be a directed graph (equivalently a network) and *S* some initial node in G chosen at random, let Bob be a "random surfer" who starts from *S* and randomly choses an outgoing edge and moves to the other endpoint of that edge; he keeps doing that without ever moving backwards. Eventually Bob gets tired and starts over from some random node, the probability of Bob not getting bored at each node and continue surfing is that node's damping factor denoted with *d*. The ranking of the nodes of *G* is the probability of Bob visiting that node (higher probability means higher rank).

PageRank problem and solution

Now that we have a fairly good understanding of the ranking method, we can present the formal definition of the PageRank algorithm

Let *A* be a node in some graph *G* with n vertices, and let T_1, T_2, ..., T_k be vertices having outgoing edges to *A*, so *A* has incoming edges from T_1, T_2, ..., T_k, also let C(v) denote the number of outgoing edges from *v*, and d(A) the damping factor of *A*, (since damping factor is usually constant we just call it *d* and in the original paper it is recommended to set it to 0.85, one reason for this choice is speedy convergence of the power method, another reason being the closeness to the reality of a web surfer), the PageRank of *A* is defined as follow:

$$PR(A) = \frac{(1-d)}{n} + d(\frac{PR(T_1)}{C(T_1)} + \ldots + \frac{PR(T_k)}{C(T_k)})$$

There is a (1-*d*) chance that Bob was bored at the last vertex so he started over randomly and then following that he chose vertex *A* by a $\frac{1}{n}$ chance. Or he did not get bored and entered *A* from one of {T_1, ..., T_n} since T_i has C(T_i) outgoing edges the value of its links (votes) is divided between them.

Note that the described formula gives a probability distribution over the vertices of *G* so $\sum_{v \in G} PR(v) = 1.$

Also to make life easier for any vertex that doesn't have any outgoing edges we assume that it has an outgoing edge to every vertex in *G* including it self, since when Bob reaches this vertex he has no option other than choosing a vertex at random and continuing.

Now if we look closely we can see that the above definition can be written for all vertices as a at once as a vector by matrix multiplication.

Let H = ($h_{i,j}$) where $h_{i,j} = \begin{cases} \frac{1}{L(v_j)}, & v_j \text{ is connected to } v_i \\ 0, \text{otherwise} \end{cases}$ it is easy to see that this definition of *H* is equivalent to $H = (D^{-1}A)^T$ where *D* is the diagonal out-degree matrix and *A* is the basic adjacency matrix.

Let PR = (PR(v_1), PR(v_2), ..., PR(v_n)).

And finally let *e* be an n-vector with all of its entries equal to 1.

Now we define the google matrix as $G = dH + ((1-d)e \times V^{(0)})$ where $V^{(0)}$ is personalization vector that defines the probability distribution of selection of the starting vertex, $V^{(0)}$ is initially set to $(\frac{1}{n}, \frac{1}{n}, \ldots, \frac{1}{n})$.

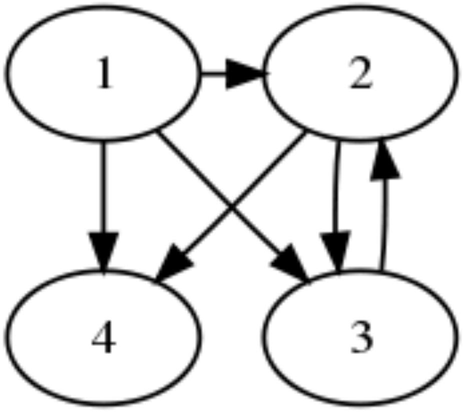
EXAMPLE:

A

0	1	1	1
0	0	1	1
0	1	0	0
1	1	1	1

H

0	0.33	0.33	0.33
0	0	0.5	0.5
0	1	0	0
0.25	0.25	0.25	0.25



D

3	0	0	0
0	2	0	0
0	0	1	0
0	0	0	4

D_inv

0.33	0	0	0
0	0.5	0	0
0	0	1	0
0	0	0	0.25

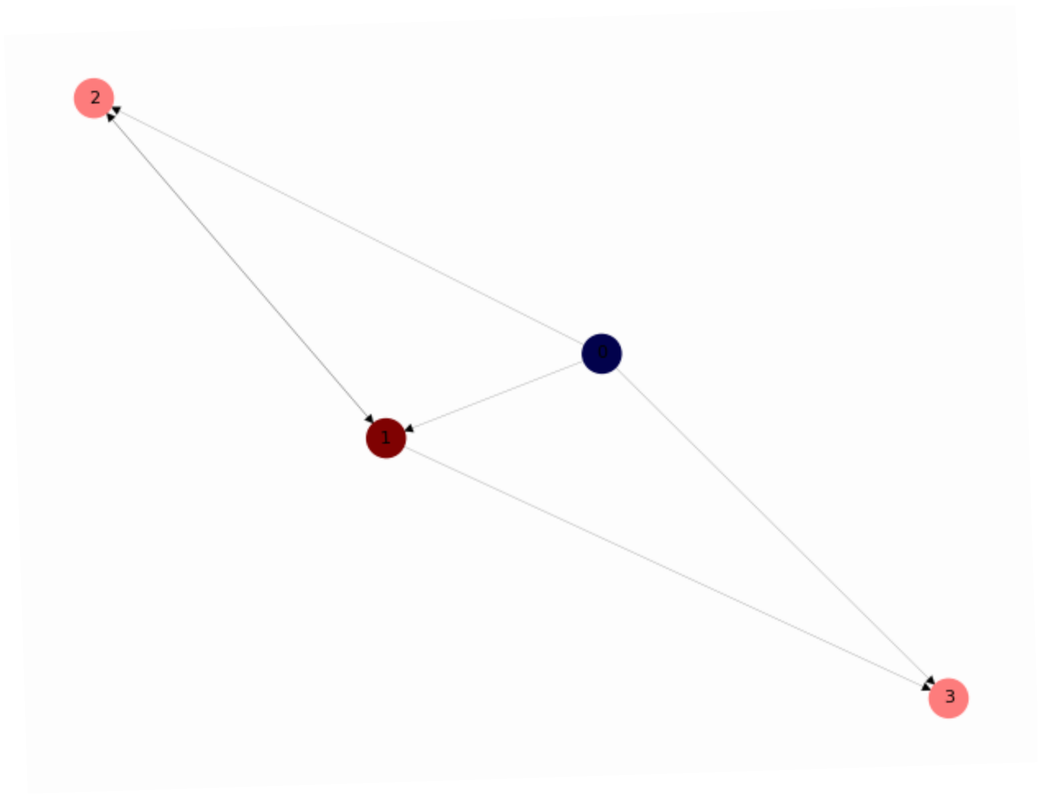
G

0.0375	0.318	0.318	0.318
0.0375	0.0375	0.4625	0.4625
0.0375	0.8875	0.0375	0.0375
0.25	0.25	0.25	0.25

vG = v, therefore v is the eigenvector corresponding to eigenvalue 1.

Using any method (i.e. inverse power method) the eigenvector is calculated: v = (0.0967, 0.3606, 0.2712, 0.2712)^T so the PageRank method generates the following ranking:

Index	Rank
0	**1**
1	4
2	3
3	3



Since *G* is defining the probability distribution of Bob's movements so the sum of each row of *G* is 1 therefore *G* is a stochastic (probability) matrix.

Then we can see that:

$$PR^{(k)} = PR^{(k-1)} \times G \qquad \text{(I*)}$$

PR$^{(0)}$ is initially is set to normal distribution $(\frac{1}{n}, \frac{1}{n}, \ldots, \frac{1}{n})$.

The above method gives an easy iterative solution to the PageRank problem, we can continuously calculate PR$^{(k+1)}$ until ||PR$^{(k+1)}$ - PR$^{(k)}$|| ≤ ε. The original paper stated that around 50 iterations give an acceptable approximation for a network with 300 million nodes. However since *G* is stochastic (I*) is a Markov chain therefore the exact solution to PageRank problem can be calculated.

Note: the $V^{(0)}$ vector and all the other places that we have used normal distribution can be replaced with other probability distribution, in fact a very useful and simple variation of the PageRank algorithm is the personalized PageRank algorithm that uses (1, 0, ..., 0) as the initial distribution to simulate always starting from one specific vertex as if Bob resided in that vertex, therefore the PageRank vector resulted from this algorithm indicates the probability of reaching each vertex starting from that specific starting vertex. This variation is used in recommendation algorithms based on PageRank method.

The PageRank problem is proven to have low condition number and the solution with power method is proven to be stable and more robust than similar methods like the HITS algorithm.

For detailed discussion on stability and condition number of the PageRank problem and the presented solution go to reference #1.

Changes to the original paper

One of the key differences of the algorithm described above from the original PageRank problem is the normalization over the outgoing degree of the vertices.

Other changes has been made to the original algorithm, but the changes that google has actually implemented in the algorithm is not known, since the it is patented and closed-sourced.

A lot of academic research has been done on this topic and centrality in general, especially eigenvector centrality variations; including randomized algorithms, parallel computable methods, localized algorithms that do not require all of the web-graph to calculate ranking and are updatable for minor changes in the network.

Stochastic (Probability) matrix

A (left) stochastic matrix is a real non-negative square matrix that the sum of each of its rows is equal to 1.

Let M be a stochastic matrix, it is easy to see that one of its right eigen pairs is (1, e), since the sum of each rows is 1. Also using the Gershgorin circle theorem we can prove that the spectral radius of M is 1, therefore 1 is the dominant eigen value of M, therefore the power method will generate 1 also the inverse power method will generate an eigenvector corresponding to the eigenvalue 1.

A stationary probability vector π for M, is a probability distribution such that it doesn't change by applying the probability matrix that is:

$$\pi \times M = \pi$$

So π is a left eigenvector corresponding to eigenvalue 1. However stationary probability vectors don't generally need to be unique, but they are unique when the probability matrix is irreducible (or more specifically: with strictly positive entries). A stochastic matrix is irreducible if and only if the corresponding weighted directed graph is strongly connected.

The matrix G that we previously constructed has strictly positive entries, since at any vertex Bob could get bored and randomly chose any other vertex to continue his traversal. Therefore any two vertices could be visited consecutively, consequently the Markov chain formed by G has a unique answer.

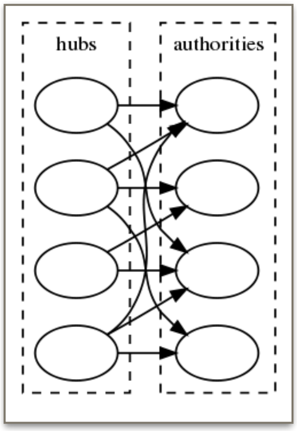
So that unique stationary vector is the solution to the PageRank problem and could be easily calculated using the inverse power method and the error would be very small since we already know the exact corresponding eigenvalue is 1.

HITS algorithm

The PageRank algorithm includes only the incoming edges in the importance measure of a node, therefore a node that has many outgoing edges to important nodes but not many incoming edges called a hub (for example search engine or a recommendation website) would not get a high rank using the PageRank method.

The HITS (hypertext induced topics search) is another ranking method similar to PageRank however instead of generating a single ranking vector, generates two vectors, one the authority vector and the other one the hubness vector.

The authority vector is very similar to the PageRank vector as it shows how much good nodes have links to a node and measures the content of a page, where the hubness vector indicates the quality of the outgoing links from a page.



A hub is node with edges to good authorities and an authority is node with edges from good hubs. We can see from the definition the the algorithm is a recursive algorithm and the hubness vector updates the authority vector and vice-versa.

We denote the hubness vector with *h* and the authority vector with *a*. Then we have $h = Aa, x = A^Th$, so we can easily see:

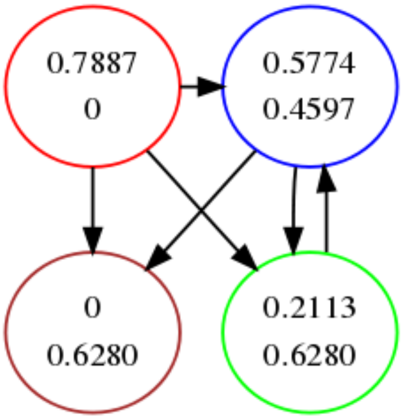
$$a = A^TAa \text{ (I)}$$

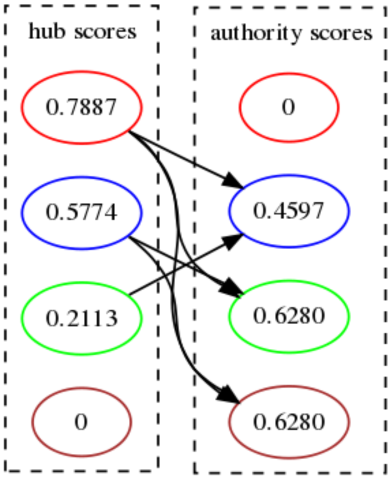
$$h = AA^Th \text{ (II)}$$

From the above equations it follows that *a* is a right eigenvector of A^TA and *h* is a right eigenvector of AA^T . Now immediately the singular value decomposition of A comes to mind, $A = U\Sigma V^T$ so $A^TA = V\Sigma^TU^TU\Sigma V^T$ and $AA^T = U\Sigma V^TV\Sigma^TU^T$, since $\Sigma^T = \Sigma$, and *V* and *U* are orthogonal. $A^TA = VSV^T$ and $AA^T = USU^T$

where *S* is the diagonal matrix of eigenvalues of *A*. The first vectors of U and V are the first eigenvectors of AA^T and A^TA respectively which are the same as *h* and *a*.

EXAMPLE:





Adjacency matrix

	1	2	3	4
1	0	1	1	1
2	0	0	1	1
3	0	0	0	0
4	0	1	0	0

Eigenvalues

2.1753	1.1260	0	0

U

0.7886	0.2113	-0.5773	0
0.5773	-0.5773	0.5773	0
0	0	0	1
0.2113	0.7886	0.5773	0

V

0	0	0	1
0.4597	0.8880	0	0
0.6279	-0.3250	-0.7071	0
0.6279	-0.3250	0.7071	0

So having the SVD of the adjacency matrix of a network, we can calculate the HITS ranking of the network.

Centrality

In graph theory and especially network analysis, an important measure on the structure of the graph is the vertex centrality.

Importance (centrality) of a vertex *v* is defined by connections of *v* to other important (central) vertices and vice-versa. This definition is not formal and too recursive because the purpose is to introduce an idea not a method, this idea of ranking vertices and assigning importance to them enables us to measure how much of an impact each vertex has on the network and using this method we can solve resource allocation problems efficiently, for example vaccine distribution and infection spread can be modeled using centrality.

There are multiple specific measures of centrality, for example the PageRank algorithm that we have discussed above is a centrality measure that uses infinite random walks with teleportation, Katz centrality is another centrality measure that employs infinite random walks, but it penalizes distance. Both of these algorithms are closely related to eigenvector centrality, which is the general case of infinite walk centrality measures.

Eigenvector centrality

The eigenvector centrality for a vertex v in a *graph* G is denoted by x_v and it's defined as follow :

$$x_v = \frac{1}{\lambda} \sum_{u \in G} a_{v,u} x_u$$

It is then easy to see that the eigenvector centrality vector denoted by *x* including all x_i's follows $Ax = \lambda x$ where *A* is the adjacency matrix of *G* therefore x is a right eigenvector of A. There are many solutions to *x* however it is desired that all entries of *x* be non-negative (since any vertex should be visitable and x is a probability distribution), so by the Perron-Frobenius theorem the appropriate solution to x is the dominant eigenvector, therefore eigenvector centrality could be calculated using the power method.

If we look at the definition presented above we can see that by adding some notion of normalizing over outgoing edges and implementing some teleportation mechanism the PageRank method can be derived from the general eigenvector centrality.

Infection spread modeling

As mentioned before centrality rankings are random walk analysis on a network, and the spread of an infectious disease is also modeled by a random walk on a general undirected graph, since some initial set of nodes (people) are infected by the disease and other nodes adjacent to them could contract the infection from them by some probability *ß*, and there is a *cure* vector C so that at the time *t* the i_{th} node heals with a probability C_i. We assume that initially the subset S of vertices are infected and we use *s* as the indicative vector.

Since the infection spread is a random walk, a Markov chain can be generated.

Let pr(d, v) be the Markov process with the damping factor d and the personalized vector v. Using the Cheeger ratio that describes the bottlenecked-ness of a subset of vertices of a graph and the PageRank algorithm (any centrality measure) an analysis of the speed of infection and the best distribution of antidotes (vector C) can be found. For more detailed discussion on infection spread modeling go to reference #4.

References

1) Internet Mathematics Vol. 1, Deeper inside PageRank.

Amy N. Langville, Carl D. Meyer.

2) The anatomy of a large-scale hypertextual Web search engine. Sergey Brin, Lawrence Page. Computer science department, Stanford university.

3) MATHEMATICS BEHIND GOOGLE'S PAGERANK ALGORITHM. Brian Moore, 2018.

4) Distributing antidote using PageRank vectors. Fang Chung, Paul Horn, Alexander Tsiatas.

5) https://en.wikipedia.org/wiki/PageRank

6) https://en.wikipedia.org/wiki/Centrality

7) https://en.wikipedia.org/wiki/HITS_algorithm

8) https://computerscience.chemeketa.edu/cs160Reader/_static/pageRankApp/index.html (PageRank calculator).

9) https://snap.stanford.edu/data/ (Datasets are available here)