Dipartimento di Ingegneria Informatica, Automatica e Gestionale
Università degli Studi di Roma "La Sapienza"

Social Networks and Online Markets

MEASURING IMPACT VIA PAGERANK IN POP MUSIC

Flavia Monti

21-06-2020

Contents

1 Introduction

In this document is described the method of PageRank. PageRank is a ranking algorithm used to help search engines to produce reasonable answers. It was proposed by Sergey Brin and Lawrence Page in 1998 to rank pages, they invented it together with Google search engine.

PageRank is a famous ranking algorithm used by Google together with other factors to produce a ranking on a search. PageRank algorithm run over the Web graph. The Web is seen as a big graph where pages are nodes and links between pages are edges. The algorithm produce a ranking among all the nodes, a page will have high rank if it is linked by other high ranked nodes so the rank of a page is recursively determined by ranks of other pages.

2 History and background

In 1995 Sergey Brin and Lawrence Page were two PhD students at Stanford University and they were working on a web search engine project called Google, their objective was to improve the quality of web search engines. They were trying to construct a search engine that takes into account the structure of the Web.

Until that moment web search engines worked only on queries of the user and on the content of the page, not considering the internal structure of the web. These methods have some limitations because pages can cheat them by putting relevant words in an appropriate location inside the page so that they can be ranked higher than others which instead are more authoritative.

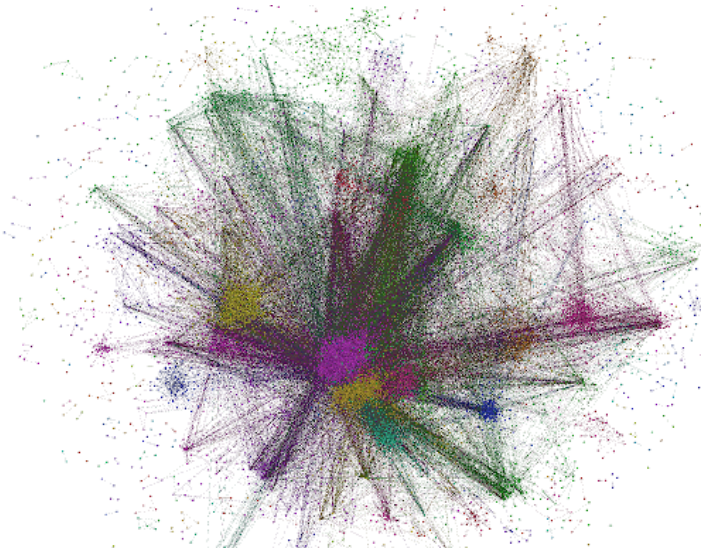


Figure 1: Web graph

Brin and Page wanted to incorporate the importance of a page to this method used until that moment. The new method that they developed works with the concept that they can construct a hierarchy of pages based on hyperlinks. Pages in the web constitute the nodes of a graph and links between pages constitute the edges of that graph, the Web is represented as a big directed graph. Links between pages are like citations in papers, they are relevant to evaluate the importance of a publication. The method assign to each page one number representing a rank and it is based on an analysis of the structure of the graph. PageRank is independent from the query but have an issue related to the pages that do not link to any other pages, called *Dangling* nodes.

PageRank's thesis is that a page is important if it is pointed to by other important pages.

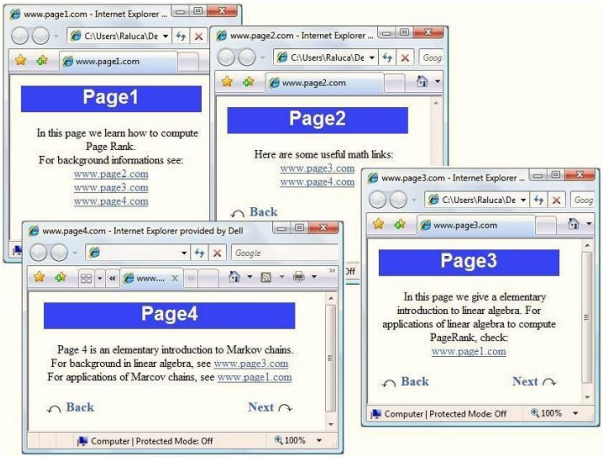
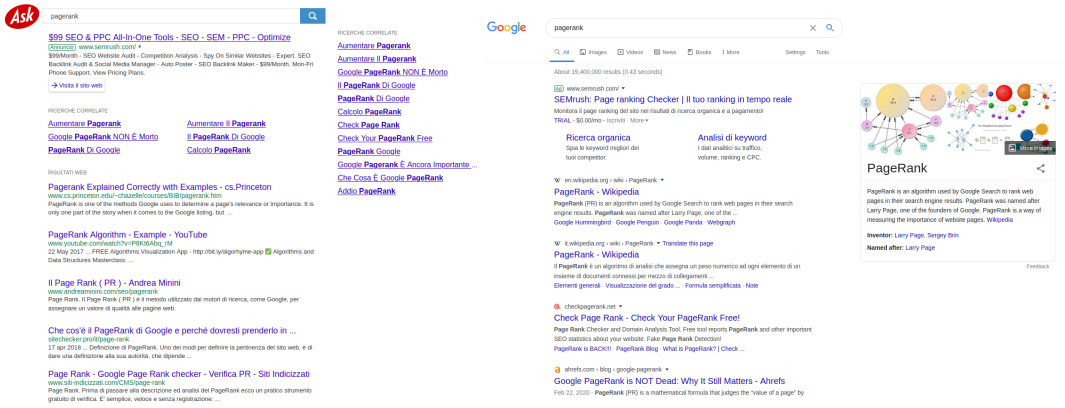


Figure 2: Web pages

In that period also another guy was working on something similar to PageRank, he was the scientist Jon Kleinberg and he was working on his project called HITS (hyperlink-induced topic search). He defined hubs and autorities, hubs are pages that contains many outlinks while a page is an authority if it has many inlinks. His method assigns two numbers to a page, a weight of authority and a weight of the hubs. HITS is query dependent, in fact scores as calculated only after acquiring the neighborhood graph according to the query.
HITS's theses are the following: a page is a good hub, and has a good hub score, if it points to good authoritative pages; and a page is a good authority if it is pointed to by a good hubs.
Kleinberg did not try to implement HITS and to build a company like Google, but some people do that and build a similar algorithm as search engine of Teoma (acquired then by Ask.com).
The two models are similar and we don't know if one influenced the other or vice-versa. However PageRank had an important dominance on the other and Google had a huge success.



(a) Ask.com

(b) Google.com

Figure 3: Search of "pagerank" in Ask.com and in Google.com

Many other ranking algorithms exist like SALSA (Stochastic Approach for Link-Structure Analysis), weighed PageRank, reverse PageRank, distance rank algorithm and topic sense algorithm. SALSA is a probabilistic extension of HITS and combine both HITS and PageRank, Weighted PageRank is an extension of PageRank and assigns rank depending on the importance of inlinks and outlinks and also on the popularity of the page. Reverse Pagerank works like PageRank but instead of using the standard graph used the reverse graph, so links are inverted. Distance rank algorithm take into account the distance of the pages and with PageRank assigns rank to the pages. Topic sense algorithm calculate the rank depending on the content of the pages.

All of these algorithms are ranking algorithms developed from 1998 to nowadays, but PageRank (1998) is the more accurate and more powerful.

3 The method

PageRank works on links. Links are the most important aspect to consider and rank is calculated depending on how they connect pages.

Initially Brin and Page defined the PageRank $R'(u)$:

$$R(u) = c \sum_{v \epsilon B_u} \frac{R(v)}{|N_v|}$$

where $B(u)$ is the set of pages linking to u and $|N_v|$ is the number of outlinks of page v.

The equation is the sum of the PageRank of all the pages backlinking to u so it is recursive but it can be computed starting from a set of ranks and iterating the operations until the value of PageRank converges. Typically the set of starting ranks are initially set to $R_0(u) = 1/n$ for all pages u.

This formula can be represented with matrices removing $\sum$ symbol and computing a $1 \times n$ vector $\boldsymbol{\pi}^T$ containing the PageRanks.

A matrix H will be such that $H_{uv} = 1/|N_u|$ if there is an edge between u and v and $H_{uv} = 0$ otherwise.

PageRank is computed as follow:

$$\boldsymbol{\pi}^T_{k+1} = \boldsymbol{\pi}^T_k H$$

Matrix H can present a problem, in fact there can be rows containing all zeros. This happens when there are pages that do not links to any other pages. These nodes are called *dangling* nodes and when there are such nodes rank is not distributed. A different problem like this is the fact that there can be cycles and the rank do not converge.

In Brin and Page paper they introduce the notion of *random surfer* model. This aspect is related to the Markov chain even if they didn't mention it. A *random surfer* represents the behaviour of a user navigating on internet, in fact the *random surfer* clicks on successive links at random. If it happens that the *surfer* is real and is in a cycle, it is unlikely that he will continue in the cycle forever but instead he would decide to jump to another link and so to a different page.

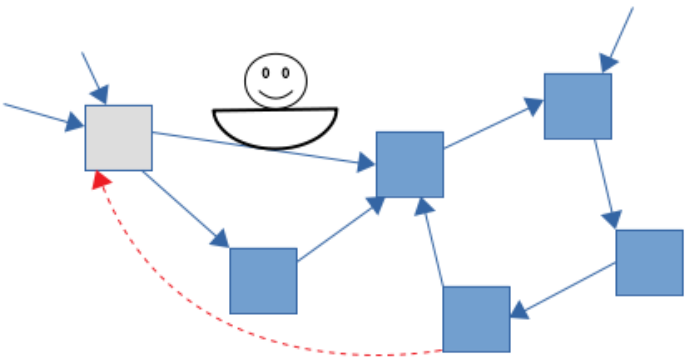


Figure 4: *Random surfer* model

In the Figure 4 it is possible to see how a surfer can be in a loop and can decide to jump to another page, following red dotted line.

To fix the problem of the *dangling* nodes and the loop problem of the *random surfer*, Brin and Page made some modification to the formula.

The $\mathbf{0}^T$ rows, that correspond to the *dangling* nodes, were replaced with $1/n\mathbf{e}^T$. The new matrix is

$$R = H + a(1/n\mathbf{e}^T), \tag{1}$$

where $\boldsymbol{a}$ is a vector having component equal to 1 if the corresponding node is a dangling node and 0 otherwise. While he matrix H seems stochastic but there can be cases in which this not happens because of the *dangling*nodes, now R is stochastic.

Taking as example the graph in Figure 5, the operation of adding $a(1/n\mathbf{e}^T)$ to the matrix H can be seen as adding the red links. *siteD* is a dangling node and is connected to each node in the graph.

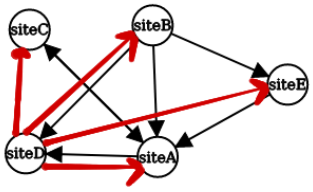


Figure 5: *siteD* dangling node problem

Another modification to the matrix is done by adding a new factor

$$R^* = \alpha R + (1-\alpha)1/n\boldsymbol{e}\boldsymbol{e}^t,$$

where α is a scalar between 0 and 1.

This modification is related to the surfer that in a certain point can decide to type a URL and to jump to a new page not following the hyperlink structure of the web.

Brin and Page decide to put $\frac{1}{n}\boldsymbol{e}\boldsymbol{e}^T$ where n is the number of nodes. Rather than use this value it is possible also to use a generic vector $v^T > 0$ that is called personalized vector.

Matrix R^* can be rewrite by applying the definition of matrix R

$$\begin{aligned} R^* &= \alpha R + (1-\alpha)1/n\boldsymbol{e}\boldsymbol{e}^t \\ &= \alpha(H + a(1/n\mathbf{e}^T)) + (1-\alpha)1/n\boldsymbol{e}\boldsymbol{e}^t \\ &= \alpha H + (\alpha a + (1-\alpha)\boldsymbol{e})1/n\boldsymbol{e}^t. \end{aligned} \tag{2}$$

The method can be seen as a Markov chain where each state depends on the previous state, PageRank of a node depends on the PageRank of the nodes linking to it. Following the hyperlink structure of the web, they built R^*, a stochastic and primitive matrix.

A stochastic matrix is a matrix used to describe the transitions of a Markov chain, each entries is a non negative real value and represent probability. Depending on the type of the matrix (right or left) the sum of the values on the row or on the column is 1, in this document it is considered the row stochastic matrix. A primitive matrix is a square non negative matrix A if there exists k such that for all i, j, entry of A_{ij}^k is positive. A primitive matrix means that the matrix is aperiodic and irreducible.

With this new matrix R^*, stochastic and primitive, a unique row vector $\boldsymbol{\pi}^T$ called PageRank exists and it is the stationary vector of the Markov chain.

Following what said before and considering the graph in Figure 6), R^* will be construct as follow.

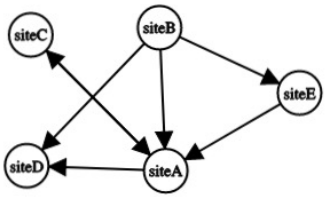


Figure 6: Example

The adjacency matrix is

$$\bar{A} = \begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array} \begin{array}{c} \begin{array}{ccccc} a & b & c & d & e \end{array} \\ \begin{pmatrix} 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}.$$

Matrix H is

$$H = \begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array} \begin{array}{c} \begin{array}{ccccc} a & b & c & d & e \end{array} \\ \begin{pmatrix} 0 & 0 & 1/2 & 1/2 & 0 \\ 1/3 & 0 & 0 & 1/3 & 1/3 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}.$$

H has a row containing all zeros that are substituting with $\frac{1}{n}\boldsymbol{e}^T = \frac{1}{5}\boldsymbol{e}^T$ obtaining

$$R = \begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array} \begin{array}{c} \begin{array}{ccccc} a & b & c & d & e \end{array} \\ \begin{pmatrix} 0 & 0 & 1/2 & 1/2 & 0 \\ 1/3 & 0 & 0 & 1/3 & 1/3 \\ 1 & 0 & 0 & 0 & 0 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}.$$

Matrix R is stochastic, there is another step in order to have a also the primitive property. R^* is computed as follow using $\alpha = 0.85$.

$$R^* = 0.85H + (0.85 \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} + 0.15 \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}) \frac{1}{5} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

$$= \begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array} \begin{array}{c} \begin{array}{ccccc} a & b & c & d & e \end{array} \\ \begin{pmatrix} 3/10 & 3/10 & 17/40 & 17/40 & 3/10 \\ 7/12 & 3/10 & 3/10 & 7/12 & 7/12 \\ 23/20 & 3/10 & 3/10 & 3/10 & 3/10 \\ 3/10 & 3/10 & 3/10 & 3/10 & 3/10 \\ 23/20 & 3/10 & 3/10 & 3/10 & 3/10 \end{pmatrix} \end{array}.$$

It is possible to compute the PageRank vector $\boldsymbol{\pi}^T$, that is the stationary vector of R^*.

4 Computation of PageRank

The computation of the PageRank is eigenvector problem:

$$\begin{cases} \boldsymbol{\pi}^T R^* = \boldsymbol{\pi}^T \\ \quad \boldsymbol{\pi}^T \boldsymbol{e} = 1 \end{cases}$$

$\boldsymbol{\pi}^T \boldsymbol{e} = 1$ is a normalized factor needed to ensure that $\boldsymbol{\pi}^T$ is a probability vector.

The solution of this problem is the normalized dominant left-hand eigenvector of R^*. R^* is a stochastic and primitive matrix and its dominant eigenvalue is equal to 1.

One way to find this eigenvector is calculating eigenvalues and eigenvectors and then normalize eigenvectors by dividing them by them sum. There is a problem with this method, when the matrix is very large the computation is not so efficient.

Another way to find the dominant eigenvector is by applying the power method as mentioned by Brin and Page in their paper. This method is a numerical method that iteratively finds the dominant eigenvector of a matrix. The fact that it finds the dominant eigenvector of a matrix means also that it finds the stationary vector of a Markov chain.

PageRank can be computed as follow:

$$\boldsymbol{\pi}_{k+1}^T = \boldsymbol{\pi}_k^T R^*.$$

At every loop $\boldsymbol{\pi}_{k+1}^T$ vector is computed and a check for convergence is done, making a comparison with the previous result $\boldsymbol{\pi}_k^T$. If the difference between the two is small the iteration can stop, otherwise it continues.

The pseudo-code of this method is the following:

Pseudo-Code 1: Power method

$\boldsymbol{\pi}_0^T = initial\_vector$;
do
  $\boldsymbol{\pi}_{k+1}^T = \boldsymbol{\pi}_k^T R^*$;
  $\delta = ||\boldsymbol{\pi}_{k+1}^T - \boldsymbol{\pi}_k^T||_1$;
while $\delta > \tau$;

$\boldsymbol{\pi}_0^T$ is the initial vector to compute PageRank and it is usually set to $1/n\boldsymbol{e}^T$.

The main operation in the algorithm is $\boldsymbol{\pi}_{k+1}^T = \boldsymbol{\pi}_k^T R^*$ and knowing that R^* can be written as $R^* = \alpha(H) + (\alpha \boldsymbol{a} + (1-\alpha)\boldsymbol{e})1/n\boldsymbol{e}^t$, the formula can be reformulated as:

$$\begin{aligned} \boldsymbol{\pi}_{k+1}^T &= \boldsymbol{\pi}_k^T R^* \\ &= \alpha \boldsymbol{\pi}_k^T (H) + (\alpha \boldsymbol{\pi}_k^T \boldsymbol{a} + 1 - \alpha) 1/n\boldsymbol{e}^T, \end{aligned}$$

note that in the formula there is not $\boldsymbol{\pi}_k^T \boldsymbol{e}$ because $\boldsymbol{\pi}_k^T \boldsymbol{e} = 1$, $\boldsymbol{\pi}^T$ is a probability vector.

In the equation there is a sparse matrix H, this means that the each multiplication between matrix H and vector $\boldsymbol{\pi}_k^T$ can be computed in $O(nnz(H))$, where $O(nnz(H))$ is the number of non zero element of the sparse matrix H.
The execution of the power method require $O(nnz(H))$ times the number of iterations.

Making a comparison between the power method and other iterative methods, like for example Jacobi or Gauss-Seidel, power method is the slowest but there are some characteristics that make it the most used. It is simple and the implementation is very basic. It uses a sparse matrix H instead of a dense matrix that can be sometimes very difficult to manipulate. Another characteristic is the fact that the coefficients of the matrix are not modified and their are accessed only by vector-matrix multiplication. There are only three elements that need to be stored: the matrix H, the vector $\boldsymbol{a}$ containing 0 and 1 depending on dangling nodes, and the vector $\boldsymbol{\pi}^T$ storing PageRank of each nodes.

Parameters

Many factors influence the PageRank, such as the values assigned to the parameters α and $\frac{1}{n}\boldsymbol{e}^T$, the row vector that replace $\mathbf{0}^T$ in the matrix H. All these parameters can alter the result and the number of iterations of the method.

The value assigned to α is 0.85 in most of the cases and the reason is because with this value the power method quickly converge to PageRank vector. Using that value the method needs only 114 iterations for convergence with tolerance 10^{-8}, if for example $\alpha = 0.99$ the number of iterations to achieve same tolerance 10^{-8} are 1833. The lower values of α brings faster convergence. Another way to see that value is considering the equation of the matrix $R^* = \alpha R + (1-\alpha)1/n\boldsymbol{e}\boldsymbol{e}^t$ and a web surfer clicking on likns. Here $\alpha = 0.85$ implies that $0.85 \approx \frac{5}{6}$ of the time the surfer randomly click on links and $(1-0.85) \approx \frac{1}{6}$ of the time the surfer will type the address of a new page to jump in, this behaviour seems like the reality. If instead $\alpha = 0.99$, this gives more power to the hyperlink structure of the web and not on the problem of dangling nodes.
Choosing different value of α modifies the results on the PageRank vector only a bit, the top value remains the same in most of the cases while going down to the lowest values it is possible to see some differences.
$\frac{1}{n}\boldsymbol{e}\boldsymbol{e}^T$ is the factor used to reflect the behaviour of the surfer that is navigating the web following the hyperlink structure, deciding then to jump to a specific page by typing the URL.
Rather than using this factor, another vector can be used $\boldsymbol{e}\boldsymbol{v}^T$, where $\boldsymbol{v}^T$ is a probability vector called personalization vector, the use of this new vector change the name of PageRank into personalized PageRank. Using $\boldsymbol{e}\boldsymbol{v}^T$ instead of $\frac{1}{n}\boldsymbol{e}\boldsymbol{e}^T$ means that the probabilities are no longer uniformly distributed.
Matrix $R*$ becomes $R^* = \alpha R + (1-\alpha)\boldsymbol{e}\boldsymbol{v}^t$ and the computation of the PageRank is

$$\begin{aligned}\boldsymbol{\pi}_{k+1}^T &= \boldsymbol{\pi}_k^T R^* \\ &= \alpha\boldsymbol{\pi}_k^T(H) + (\alpha\boldsymbol{\pi}_k^T\boldsymbol{a} + 1 - \alpha)\boldsymbol{v}^t,\end{aligned}$$

Using different personalization vectors produced different output (values) on the PageRank. Focusing on this aspect, studies were done relating to the fact either of having only one single vector equal for everyone or having different personalization vectora for each person. For example if someone is interested in sport he would have a personalization vector $\boldsymbol{v}^T$ such that components v_i related to pages inherent to sport have higher values.
An extreme case is to have $\boldsymbol{e}\boldsymbol{v}^T$ such that the probability vector is concentrated entirely on a single web page. In Brin and Page paper they reported and example with Netscape home page and the home page of John McCarthy, a famous computer scientist. They generated PageRanks from the perspective of two novice users having as home page respectively Netscape and John McCarthy. In both cases the respective home page got the highest PageRank, the following results were their immediate links. In Figure 7 are reported PageRanks of the two cases, there are many different pages, those related to computer science are ranked higher if having McCarthy as home page. For example, the Web page of a member of Stanford computer science faculty is more than six percentile points higher on the McCarthy-rank (100.00% vs 93.89%).

The usage of this personalized vector switch the caracteristics of query-independent and user-independent to query-dependent and user-dependent introducing complex computation.
Some researchers tried to create a personalized Pagerank not for each users but for group of users. In this way the computation is the same high but not so high than for each user.

Web Page Title	John McCarthy's View PageRank Percentile	Netscape's View PageRank Percentile
John McCarthy's Home Page	100.00%	99.23%
John Mitchell (Stanford CS Theory Group)	100.00%	93.89%
Venture Law (Local Startup Law Firm)	99.94%	99.82%
Stanford CS Home Page	100.00%	99.83%
University of Michigan AI Lab	99.95%	99.94%
University of Toronto CS Department	99.99%	99.09%
Stanford CS Theory Group	99.99%	99.05%
Leadershape Institute	95.96%	97.10%

Figure 7: Netscape and John McCarthy home pages PageRanks

Convergence and number of iterations

Depending on the choice taken relative to the value assigned to α and to the tolerance τ, the number of iterations of the method vary. The convergence of the method and the number of iterations are related.

Brin and Page used as termination criterion to stop when the residual is less than some predetermined tolerance level $\boldsymbol{\tau}$. This influence the convergence of the method and to analyze this aspects it is needed to make a small introduction.

In general, the rate of convergence of the power method applied to a matrix depends on the ratio of the subdominant eigenvalue λ_2 and the dominant eigenvalue λ_1. For stochastic matrices, like for example H or R^*, $\lambda_1 = 1$ and all other eigenvalues are smaller, so $|\lambda_2|$ governs the convergence $|\lambda_2| < 1$.

Before continuing in the discussion it is important to say that the primitive property of matrix is fundamental in this argument and this is one of the reasons why Brin and Page modified the initial matrix H to create a primitive stochastic matrix R^*.

An important observation on matrix R^* is relative to its definition $R^* = \alpha(H) + (\alpha\boldsymbol{a} + (1 - \alpha)\boldsymbol{e})1/n\boldsymbol{e}^t$. Given the spectrum of R^* and H, it is proved that eigenvalues λ_k for $k = 1, ...1$ of R^* are related to the eigenvalues μ_k for $k = 1, ..., n$ of H in that way $\lambda_k = \alpha\mu_k$ for $k = 1, 2, ..., n$. Given the structure of the graph is very likely that $|\mu_2| = 1$ and so $|\lambda_2| = \alpha$.

The asymptotic rate of convergence of the power method is the rate at which $\alpha^k \to 0$.

An expander graph is a graph where every subset of nodes S has a neighborhood that is larger than some factor α times $|S|$, this factor is called expansion factor.
A graph is said to have a good expansion factor if and only if the largest eigenvalue is sufficiently larger than the second largest eigenvalue.
This last analysis seems strictly related to what said before.

A random walk on a graph is a stochastic process where at any step the node on the graph is analyzed and a random choice on the next step is done following a link. A random walk on a graph is rapidly-mixing (it is converging) if and only if the graph is an expander (or in other words it has eigenvalues separation), the walk quickly converges to a limiting distribution on the set of nodes of the graph.

In PageRank method the limitation of the distribution of the random walk on the graph is a very important aspect, in particular rank of a node is the limiting probability that a step in the random walk will be in that node after a large time. This is correlated to what said before.

Brin and Page have chosen a value of $\alpha = 0.85$ and measured the number of iterations needed to reach a tolerance level $\boldsymbol{\tau}$. Tolerance level is measured by the residual

$$||\boldsymbol{\pi}_{k+1}^T - \boldsymbol{\pi}_k^T||_1 = ||\boldsymbol{\pi}_k^T R^* - \boldsymbol{\pi}_k^T||_1,$$

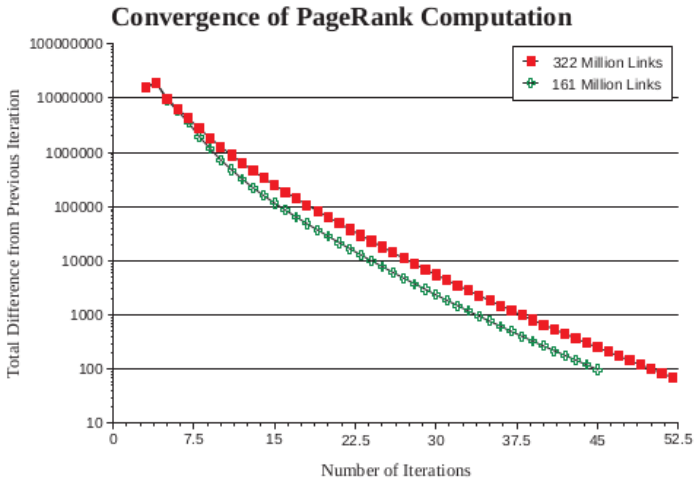


Figure 8: Convergence of full and half size edges graphs

the estimation of the number of iterations is $\frac{\log_{10}\boldsymbol{\tau}}{\log_{10}\alpha}$. Fixed $\alpha = 0.85$, if $\boldsymbol{\tau} = 10^{-6}$ there are ≈ 85 iterations until convergence to the PageRank vector, if $\boldsymbol{\tau} = 10^{-8}$ there are ≈ 114 iterations and if $\boldsymbol{\tau} = 10^{-10}$ the number of iterations are ≈ 142. Brin and Page's matrix R converges quickly using 50 to 100 power iterations, that means $\boldsymbol{\tau}$ can range from 10^{-3} to 10^{-7}.

In Figure 8 is represented the rate convergence of two graphs. The graph with 322 million links converge in 52 iterations while the graph with $\frac{322}{2} = 161$ million links converge in 45 iterations.

Storage

Every search engine requires storage to store different data like information of webpages, locations, content score information, PageRank scores and the hyperlink graph of the entire web. PageRank method requires access to matrix H and to vectors $\boldsymbol{v}^T$, $\boldsymbol{a}$ and $\boldsymbol{\pi}_k^T$.

$$\boldsymbol{\pi}_{k+1}^T = \alpha\boldsymbol{\pi}_k^T(H) + (\alpha\boldsymbol{\pi}_k^T\boldsymbol{a} + 1 - \alpha)\boldsymbol{v}^t.$$

Table in Figure 9 report all the characteristics of each element.

Entity	Description	Storage
$\mathbf{H}$	sparse hyperlink matrix	$nnz(\mathbf{H})$ doubles
$\mathbf{a}$	sparse binary dangling node vector	$\|\mathrm{D}\|$ integers
$\mathbf{v}^T$	dense personalization vector	n doubles
$\boldsymbol{\pi}^{(k)T}$	dense current iterate of PageRank power method	n doubles

Figure 9: PageRank's storage requirements

$nnz(H)$ is the number of nonzeros component in H, $|D|$ is the number of dangling nodes and n is the number of pages in the web graph. When $\boldsymbol{v}^T = \boldsymbol{e}^T/n$, no storage is required for it.
The first thing to consider is about matrix H, it is important to know if it fits in main memory, when H fits in main memory, computation of PageRank can be implemented using the standard formula, but when the H matrix does not fit in main memory computation becomes a little more complex. There are two options: compress the data needed to fits in main memory and implement a modified version of PageRank to work on it; or keep the data in its uncompressed form and develop input/output-efficient implementations of the computations that must take

place on the large data.
In most of the cases, even if the compression is not needed, some modifications are applied.
For example matrix H can be decomposed into the product of the inverse of the diagonal matrix D holding outdegrees of the nodes and the adjacency matrix A, $H = D^{-1}A$, where $D^{-1}_{ij} = \frac{1}{d_{ii}}$ if i is a nondangling node, 0 otherwise. In that way, rather than storing $nnz(H)$ real numbers, it's needed to store n integers for D and $nnz(H)$ integers for the locations of 1's in A. Storing integers require less storage than real numbers and using the vector $diag(D^{-1})$ the multiplication $\boldsymbol{\pi}^T_k H$ can be seen as $\boldsymbol{\pi}^T_k D^{-1}A = \boldsymbol{\pi}^T . * diag(D^{-1}A)$ where $.*$ represents componentwise multiplication of the elements in the two vectors. The first part requires n multiplications and since A is the adjacency matrix the total computation requires n multiplication and $nnz(H)$ additions, saving $nnz(H) - n$ multiplications. This example can be applied only on *random surfer* model, considering an *intelligent surfer* model other methods are applied such as compressed row storage or compressed column storage.
Matrices are stored as adjacency lists of the columns. In order to compute the PageRank vector, the PageRank power method requires vector-matrix multiplications at each iteration k, so given the fact that matrices are stored in adjacency lists of the column this is very powerful. Column i contains the inlinks information for page i, this information is more important than the information contained in the rows relative to outlinks.

5 Applications

PageRank was created to be used in Google search engine managing and scoring web pages using the hyperlink structure, determining which pages are more important.

All the characteristics of the algorithm make it suitable for other applications very distant from purpose for which it was invented.

Neuroscience

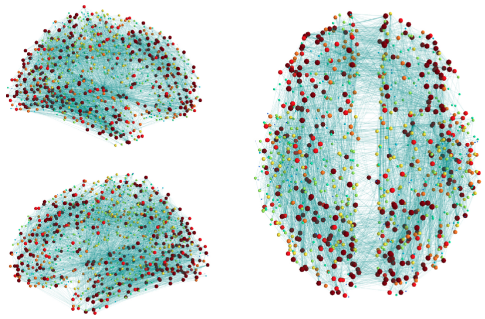


Figure 10: Human brain connectome

The human brain connectome is one of the most important networks and what is known about it is very little. Applied network theory is one of a variety of tools currently used to study it. PageRank helped evaluating the importance of brain regions, given observed correlations of brain activity. Some experiments were done in a graph like the following, two voxels of an MRI scan are connected if the correlation between their functional MRI time-series is high. Edges with weak correlation are deleted and the others are retained with either binary weights or the correlation weights. The resulting graph is undirected, the usage of PageRank combined with community detection and known brain regions gives the possibility to understand changes in brain structure across population that correlate on some factor.
PageRank output result not so goo, in other experiment using reversed PageRank the results wertr better.

Engineered systems

PageRank method can also be used to study complex engineered systems. Network analysis methods like PageRank help revealing all the details of the system.
A first example is MonitorRank by Kim et al., it is a system to provide guidance to a systems administrator or developer as they perform searching through error logs and tracing debugging information. It returns a ranked list of systems based on the likelihood that they contributed to, or participated in, an anomalous situation. Consider the systems underlying the LinkedIn website: each service provides one or more APIs that allow other services to use its resources. For instance, the web-page generator uses the database and photo store. The photo store also again uses the database, and so on. Each combination of a service and a programming interface becomes a node in the MonitorRank graph. Edges are directed and indicate the direction of function calls. Given that an anomaly was detected in a system, MonitorRank solves a personalized PageRank problem on a weighted, augmented version of the call graph, where the weights and augmentation depend on the anomaly detected. The localized PageRank vector help to determine the anomaly.
Another application of PageRank is on the Linux kernel. The Linux kernel is the foundation for an open source operating system. The kernel call graph is a directed graph with million of edges, representing dependencies between functions, PageRank produces an ordering of the most important functions in Linux. This application turns out that utility functions such as printk, which prints messages from the kernel, and memset, a routine that initializes a region of memory, have highest PageRank.

Sports

Figure 11: Ranking of National football teams

In sports ranking the usage of stochastic matrices and eigenvector ranking methods are often used. The winner network is one of the most common network used. Each team is a node in the network, and node i points to node j if j won in the match between i and j. These networks are often weighted by the score by which team j beat team i. Govan et al. used PageRank to rank football teams with these winner networks. The intuitive idea underlying these rankings is the behaviour of a random fan that follows a team A until another team B beats A so the fan pick up the new team B, and periodically restarts with an arbitrary team.

Literature

PageRank can be applied also to the literature, it answers to different questions like for example what are the most important books, which story paths in hypertextual literature are most likely

and what book should a person read next.
Jockers defined a complicated distance metric between books using topic modeling ideas from latent Dirichlet allocation. Using PageRank he argues that Jane Austin and Walter Scott are the most original authors of the 19th century.
Hypertextual literature is particular type of literature containing multiple possible story paths for a single novel. Most of the books consists of a set of storylets, at the conclusion of a storylet, the story either ends or presents a set of possibilities for the next story. The *random surfer* model for PageRank maps perfectly to how users read these books and the algorithm gives useful information about the properties of the stories.
Traditional library catalogs use a carefully curated set of index terms to indicate the contents of books. Websites like LibraryThink and Shelfari allow their users to curate their own set of index terms for books that they read and share this information among the user sites. The data on these websites consists of books and tags that indicate the topics of books. BookRank, which is a localized PageRank on the bipartite book-tag graph invented by Meng, produces suggestions for what to read next.

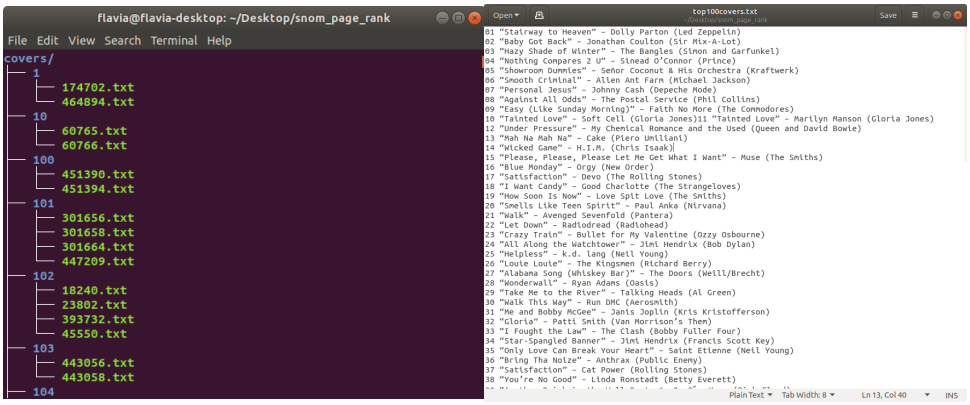
Social networks

In a social network, the nodes are people and the edges are some type of relationship. PageRank can be applied to such type of graph giving as output useful information. It can help solving link prediction problems to find individuals that will become friends soon. It has a classic role in evaluating the centrality of the people involved to estimate their social status and power. And finally it can evaluate the potential influence of a node on the opinions of the network.
Centrality methods have a long history in social networks. The first use of PageRank in a large-scale social network was the BuddyRank measure employed by BuddyZoo in 2003. BuddyZoo collected contact lists from users of the AOL Instant Messenger service and assembled them into one of the first large-scale social networks studied via graph theoretic methods.
PageRank has also been used to rank individuals in Twitter network by their importance and to help characterize properties of the Twitter social network by the PageRank values of their users.
Finding influential individuals is one of the important questions in social network analysis. This amounts to finding nodes that can spread their influence widely. More formalizations of this question result in NP-hard optimization problems. Reverse PageRank, instead of traditional PageRank, is the correct model to understand the origins of influence, the distinction is much like the usage of hubs and authorities in other ranking models like HITS.

5.1 Pop music

PageRank can be applied also to the music world. In particular it can be used to find the group or single artist that has been covered the most. The covers-graph is a graph where each node is an artist and an edge from node i to node j means that artist i has done a cover a song of artist j. Before constructing the graph, all the data need to be found. These data are collected from different websites and all information are grouped in folders o in different `.csv` files.
Data grouped in folders are structured like that: each folders is related to a song and inside a single folder there are different `.txt` files each related to an artist, if in the `.txt` file there is the sentence *'First release'* or *'First recording'* this means that the song is owned by that artist and all the other artist mentioned in the other `.txt` files have covered it.

The directed graph is generated using `networkx` and its functions, in Figure 13 there is a representation.



(a) Folders (b) `.csv` files

Figure 12: Data for covers-graph

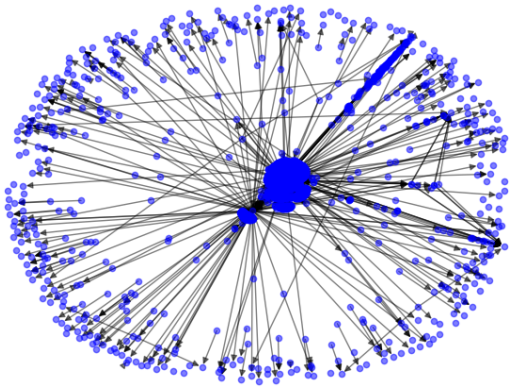


Figure 13: Covers-graph

The implementation of the PageRank algorithm follows the pseudo code *P-code* (1) on page 7 of the Power method, using the standard formula $\boldsymbol{\pi}_{k+1}^T = \alpha \boldsymbol{\pi}_k^T(H) + (\alpha \boldsymbol{\pi}_k^T \boldsymbol{a} + 1 - \alpha)1/n\boldsymbol{e}^t$.

```
1  def pagerank(g, max_iter, alpha, tau):
2    sg = nx.stochastic_graph(g)     #stochastic graph
3    n_nodes = nx.number_of_nodes(g)
4    nodes = g.nodes()
5
6    PI = [1.0/n_nodes] * n_nodes    #initialization of pagerank
7
8    a = []                          #dangling nodes vector
9    for n in nodes:
10     if g.out_degree(n):
11       a.append(1)
12     else:
13       a.append(0)
14
15   H = nx.adjacency_matrix(sg)     #stochastic matrix
16   for i in range(max_iter):
17     pi_previous = PI
18
19     v1 = [0] * n_nodes           #v1 = alpha(pi_previous^T*H)
20     for r in range(n_nodes):
21       row = H[r,:].toarray()
```

```
22        for c in range(n_nodes):
23          v1[c] += pi_previous[c]*row[0][c]
24      v1 = [alpha*v for v in v1]
25
26      dang_pi = 0                 #v2 = alpha(pi_previous^T*a)1/n*e^T
27      for e in range(n_nodes):
28        dang_pi += pi_previous[e]*a[e]
29      constant = alpha*dang_pi+1-alpha
30      v2 = [float(constant)/n_nodes] * n_nodes
31
32      for e in range(n_nodes):    #pi = v2 + v3
33        PI[e] = v1[e] + v2[e]
34
35      PI = normalize(PI)          #pi^T*e=1
36
37      delta = 0                   #check convergence
38      for e in range(n_nodes):
39        delta += abs(PI[e] - pi_previous[e])
40      if delta < tau*n_nodes:
41        return transform_pagerank(PI)
42    return transform_pagerank(PI)
```

Listing 1: PageRank implementation

In line 2 it is used the function `nx.stochastic_graph(g)`, this function is a particular function of the Python package `networkx` that construct a stochastic graph from a given graph. The adjacency matrix of this particular graph is a row stochastic graph that is used from line 15 for the computation of the PageRank vector. In line 35 a function is called to normalize the vector, PageRank it is a probability vector so all the components need to sum to 1.

The PageRank vector of the covers-graph is reported in Table 1. In the table there are the results of the first 10 artists output by the implementation 1 and by the `pagerank()` function of the Python package `networkx`. The parameters used are $\alpha = 0.85$ and $\tau = 10^{-6}$.

Table 1: PageRank of covers-graph

	PageRank		PageRank by `networkx`	
	Artist	PageRank	Artist	PageRank
1	BEATLES	0.131089	BEATLES	0.174324
2	BOB DYLAN	0.076884	BOB DYLAN	0.058624
3	U2	0.037365	MICHAEL JACKSON	0.026740
4	MICHAEL JACKSON	0.034016	U2	0.018703
5	LED ZEPPELIN	0.032358	LED ZEPPELIN	0.016141
6	BEACH BOYS	0.026401	BEACH BOYS	0.013697
7	MADONNA	0.018442	BYRON G. HARLAN	0.012248
8	MISFIT	0.017428	MADONNA	0.009296
9	QUEEN	0.008226	JOHN LENNON	0.008761
10	PAUL MCCARTNEY	0.002501	MISFIT	0.008707

The Beatles result to be the most covered band in both the algorithms. The ranks shows some differences, there are some artists that are in an higher position in an algorithm than the other.

These differences in the values are due to the factor that the implementation 1 can be not so accurate than the one done by `networkx`. Even though the formula used in the implementation is the formula presented by Brin and Page there can be some differences with the implementation inside `networkx`.

Also other applications of PageRank in music can be implemented, like for example the col-

laboration between the artists. In this case a different graph from the cover-graph need to be construct, each artist is again a node but edges have different meaning, they represent a collaboration between two artists.

6 Conclusions

PageRank is a global ranking of all web pages based on their location in the web graph structure.

In experiments, PageRank provide higher quality search results to users. It uses links and not the content of the pages, links from important pages are more significant than links from average pages.

A lot of applications for PageRank in addition to search can be analyze, these applications can be also very different from the scope for which it was invented for.

The good results with PageRank show that the structure of the graph is very important and useful to retrieve information.

References

[1] Monica Bianchini, Marco Gori, Franco Scarselli. *Inside PageRank*. University of Siena, 2000.

[2] Sergey Brin, Lawrence Page. *The anatomy of a large-scale hypertextual Web search engine*. Computer Science Department, Stanford University, 1998.

[3] Sergey Brin, Lawrence Page, Rajeev Motwani, Terry Winograd. *The PageRank Citation Ranking: Bringing Order to the Web*. Techical Report, Computer Science Department, Stanford University, 1998.

[4] David F. Gleich. *PageRank Beyond the Web*.

[5] Jon Kleinberg. *Authoritative Sources in a Hyperlinked Environment*. Proceedings of the ACM-SIAM Symposium on DiscreteAlgorithms, 1998.

[6] Amy N. Langville, Carl D. Meyer. *Deeper Inside PageRank*.

[7] Amy N. Langville, Cal D. Meyer. *Google's PageRank and Beyond: The Science of Search Engine Rankings*. Princeton University Press, 2006.

[8] https://en.wikipedia.org/wiki/PageRank

[9] https://networkx.github.io/documentation/stable/