Contents

1. INTRODUCTION

In August 1995, a NetCraft[1] study said there were 18.957 web pages. In April 1997 the same study talked about one million of websites. Nowadays there are more than 50 million of web pages a[1]nd the last 10 million were published in only 13 months. Consequently, the web search results grow more and more but users process only the first few results. By using traditional term-based techniques probably relevant information hidden in the long result lists will never be discovered by users. For this reason, last studies propose new ranking techniques based on hyperlink analysis in order to boost search performance. Among the most important algorithms are significant Hyper Search [1], PageRank [3] and HITS [2]. HITS is implemented in the CLEVER Project [4], while PageRank is the core of Google[2], the most successful search engine.

PageRank algorithm gives a global importance measure to each page on the web based on the number and importance of other pages linking to it. This importance is computed every certain time by an iterative process defined on the web graph in an off-line manner and the process each time requires more computation because of the actual web growth. However, there might be some situations where a quick estimation is needed, for example, if we are following the evolution of the PageRank of a certain page or group of pages. Other example is the localized search engines where incorporating global PageRanks can improve the quality of search results. Unfortunately local search engines do not have the bandwidth, storage capacity, or computational power to crawl of their large-scale counterparts. In both scenarios would be great a PageRank calculation that do not require such a level of computation. For this reason, in this study we estimate reasonable PageRank values by using local methods in order to minimize computation costs. The main idea is to expand a subgraph around the target node then calculate PageRank values based on the subgraph. The problematic of this approach basically resides in how big should be the generated subgraph and which criterion should be applied for including or excluding nodes in the graph. We will see later how these decisions are applied by following proposed methods in [5].

2. PAGE RANK REVIEW

PageRank, for short PR, is a numeric value that indicates how important a page is on the web. This value is hyperlink dependent since it is calculated with the information that hyperlinks provide. More concretely, a page will receive some amount of PR from all pages that is linked to and this amount will depend on the importance of these pages themselves and the numbers of the hyperlinks in them over which it is divided. For instance, if we have a web page and it is linked for an important page that links to 4 pages more we will receive 1/5 of its PR from that page but if this important page links to 30 pages we will receive 1/30 of its PR. Thus, the desirable situation to get a high PR would be to be linked by important pages with a few outlinks to other pages. PR believes in this criterion since it supposes that if a web page contains links to other pages it is affirming the importance of the pages linked to.

Based on last criterion we denote the PR as follows:

[1] NetCraft is an Internet services company that provides research data and analysis on many aspects of the Internet

[2] An example of popularity of Google is that google has been include as a verb in Oxford English Dictionary

$$PR(v) = \sum_{u \to v} \frac{PR(u)}{d(u)} \qquad (1)$$

The problem of this formulation is that we first need to know the importance of all pages linking to a certain page to determine its PR. However, firstly we do not know any PR but we know the percentage of PR that each page will share out with the pages that is linking to. Thus, we can rewrite the problem into one more mathematically familiar by using a matrix. Let us suppose a set of *n* pages. We define its connectivity matrix *G* as a square matrix of order *n* whose elements, called $g_{ij,}$ $1 \le i, j \le n$, are 1 if there is a link to page *i* from page *j,* with i ≠ j and 0 otherwise.

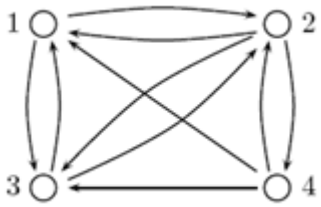


Figure 1. Directed graph

$$G = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Figure 2. Connectivity matrix corresponding to figure 1 graph

Now imagine that we surf the web at random. That is, there a person (a surfer) who is visiting a web page and randomly follows one of its links. Every link on that page has the same probability to be selected by our surfer. In our example (figure 1) if the surfer is on page two then he/she has 1/3 of probability to reach the page 1 or 3 or 4. By applying this reasoning to the rest of pages we obtain the random surfer transaction matrix *P*

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

Figure 3. Transaction matrix

The easiest way to calculate this matrix is by dividing each column of *G* over the sum of the values in *G* for that column. Notice that this only applies when this sum is different from zero. In other words, when the column does not correspond to a page without outlinks. This matrix represents the PR which is shared out between the web pages. The diagonal shows the amount of PR that is shared out between pages themselves in case of having links to themselves. Any value over

the main diagonal and its symmetric one in the lower triangular matrix denote the amount of PR that is transferred between two pages when one links to other and vice versa. To formalize this approach we define *d(i)* as the number of outlinks in a pages, called out-degree. Then we can define the matrix *P* as follows:

$$P[i,\, j] = \begin{cases} 1/d(i) & \text{if } d(i) \neq 0 \\ 0 & \text{otherwise} \end{cases} \qquad (2)$$

Notice that *P* has some special properties:

- It is a square matrix whose entries consist of nonnegative real numbers.
- If *d(i) ≠ 0* for all *i* then P [2]is a left stochastic matrix. That is, the sum of each column is one and each element takes a value between 0 and 1.

Furthermore, if we look at matrix *P* we observe other interesting properties: its eigenvalues are {1, -1/2, -1/3, -1/6}. Its spectral radius[1] is *ρ(P) = 1* and the matrix is both irreducible[2] and primitive[3]. These properties are important since the PageRank is the limit vector of probability distribution of a Markov ergodic chain. If the limit vector exists it matches with the stationary state vector and it is independent of the initial state vector. This limit state vector property applies for stochastic and primitive matrices. In our case, the matrix *P* fulfills: P ≥ 0, P is irreducible and P has only a eigenvalue (λ = 1) equals to the spectral radius *ρ(P) = 1*. The stationary vector for the matrix *P is:*

$$I = \begin{bmatrix} 0,29, & 0,32, & 0,29, & 0,10 \end{bmatrix}$$

To calculate the above vector we have used the power method. This method works by applying the following:

$$I^{k+1} = \mathbf{P}I^{k}$$

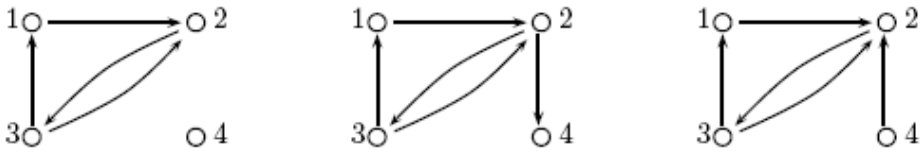
Thus, the sequence I^{k} will converge to the stationary vector *I*. Let us illustrate this convergence with the following table.

I^0	I^1	I^2	I^3	...	I^{55}	I^{56}
1	0	0,42	0,22	…	0,29	0,29
0	0,5	0,25	0,35	…	0,32	0,32
0	0,5	0,17	0,35	…	0,29	0,10
0	0	0,17	0,08	…	0,10	0,10

The stationary vector can be interpreted as follows. If the random surfer surfs through the set of linked pages (figure 1) for a while the probability to find him/her on the page 2 is 9/28. In other words, the surfer spends a 32% of his/her time on the page 2. Notice that pages 1, 2 and 3 have the same number of in-degree. The only difference between them is that the page 3 has one outlink more than page 1 and 3. Thus, page 2 share its PageRank out over 3 pages and because of this reason page 1 and 3 get less amount of PageRank from page 2.

[1] The spectral radius of a matrix is the radius of the smallest circle in the complex plane that contains all its eigen values.
[2] A matrix is irreducible if its graph is strongly connected: for each couple of nodes (i,j) with i ≠ j there is a path to reach j from i through a run of connected arcs. Otherwise, the matrix is reducible
[3] A square matrix is primitive if it is irreducible, no negative and its spectral radius is larger than either other eigen value.

In our example (figure 1) we have a matrix that is both stochastic and primitive. These are the properties that permit us to get the stationary vector - PageRank vector – but there are some cases where the link structure does not generate a stochastic and primitive matrix. Let us study the following scenarios:



a) b) c)

The situation a) does not affect the results since page 4 has both in-degree and out-degree equal to zero so it will not be present in the matrix P. The situation b) shows a dangling node. This situation appears when a page has the in-degree different to zero but its out-degree is zero. This generates a non stochastic matrix P. We will study later on how to solve this scenario. The last example, matrix c, is a stochastic matrix but in that case it is reducible since node 4 is not reachable. In the next section we will study how to solve the issues in matrix b) and matrix c) in order to get a stochastic and primitive matrix to ensure the stationary vector can be calculated.

Dangling nodes

The initial model tries to calculate the PageRank vector from matrix *P* (2). This method assumes that matrix is both stochastic and primitive. However, the matrix *P* becomes in a non stochastic one if there are dangling nodes in the graph. In this case the stationary vector will tell us nothing about the importance of every page. The following example shows this situation:

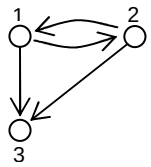


Figure 4. Graph with dangling node

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

Figure 5. Matrix P corresponding to figure 4 graph

I^0	I^1	I^2	I^3	...	I^{1076}
1	0	0.25	0.25	…	0
0	0.5	0	0	…	0
0	0.5	0	0.25	…	0

We observe how page 4 takes some of the importance from page page 2 in the step 2 but it does not pass it on to any other page. This has the effect of draining all the importance from the web. We can imagine that there are many dangling pages in the real web we want to study so we should get rid of them.

How is it possible to prevent this situation? If we can guarantee that all dangling pages have at least one link this issue can be overcome. The problem is that we cannot create randomly links for those pages since the PageRank would be falsified by increasing the importance of linked pages through non-existent links. To solve this situation we will replace the column of zeroes corresponding to a dangling node with a column in which each entry is 1/*n*, where n is the total amount of nodes. In this way although we create non-existent nodes, the PageRank of dangling nodes is shared out with all pages (even itself). The PR vector will be equally increased for all elements. In other words, the results meaning will not be altered. Furthermore in the real web graph this amount of PR is insignificant due to the huge amount of pages that it contains (approx. 25.000.000.000). The new distribution matrix is as follows: $S = P + A$ where A is:

$$A[i, j] = \begin{cases} 1/n & \text{if dangling node} \\ 0 & \text{otherwise} \end{cases}$$

Let us see how this applies to our example.

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

Figure 6. Matrix *S* corresponding to figure 4.

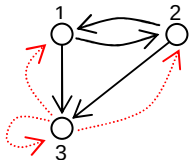


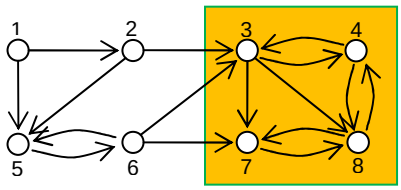
Figure 5. Graph with dangling node issue fixed

I^0	I^1	I^2	I^3	...	I^{55}
1	0	0.416	0.222	...	0.285
0	0.5	0.166	0.347	...	0.285
0	0.5	0.416	0.430	...	0.428

Notice that node 3 is getting ½ of PageRank from both nodes 1 and 2. The rest of PageRank for the node 1 corresponds with the 1/3 of PageRank generated by the link that points to itself. This amount will be also shared with the other nodes.

A final review

Even after overcoming the dangling nodes issue, our matrix is already stochastic, we cannot still guarantee that our matrix will have a stationary vector. This is because of the fact that the web structure can generate non primitive *S'* matrices. Looking at the web these matrices can appear when there are group of pages with on entry point but no exit point. The following figure illustrates this scenario:



In this example there are not dangling nodes but we can observe that links come into the orange box, but none go out, causing the same effect as dangling nodes. The graph above generates a reducible matrix since the graph is not strongly connected – for example, there is not a path to reach node 1 from node 8 -. To find a new primitive matrix we will modify the way our random surfer moves through the web. Thus far the movement of the random surfer is determined by *S* so in the above example the surfer will not be able to get out of the orange box if he/she reaches it. This behavior can be changed by adding a dampening factor α -between 0 and 1 -. With this factor the surfer is guided by *S* with probability α and he/she chooses the next page at random with probability 1 – α. Thus, the surfer will never get stuck in case of reaching the orange box. By applying this approach to our matrix *S*, we obtain the Google matrix:

$$G = \alpha S + (1 - \alpha)ve^T$$

where $0 < \alpha < 1$, $e^T \in \mathbb{R}^{1xn}$ is a ones vector $e^T = [1, 1, \ldots 1, 1]$ and $v \in \mathbb{R}^{nx1}$ is a probability vector and usually is $v = \frac{1}{n}e$. The multiplication ve^T is a matrix of order *n*. Notice that G is a stochastic since it is a combination of stochastic matrices so we can still guarantee absence of dangling nodes and furthermore G is also a primitive matrix. Thus, now it is possible to affirm that G has a unique stationary vector *I* (say PageRank vector).

Another important consideration is the factor α. It usually takes $\alpha = 0.85$ since it was the value originally used by Brin and Page, PageRank creators. If we use values close to one (with $\alpha = 1$, G = S) we would obtain more realistic results but we could leave the irreducibility of our *G* matrix and make more expensive the power method since it would require more iterations.

3. NEXT WORK

Once reviewed the PageRank algorithm we will study further the three proposed methods in [5] to estimate PageRank for a target node. These methods are the following:

- Naive method
- The Simple Influence Method
- The Indegree-Base Influence Method

Then we will set up a test environment that permits us to create a web graph at random and run the three methods in order to evaluate the results. At this point, we should be able to find an improvement of some of the available algorithms or at least come up with some suggestions to help PageRank estimation easier.

4. WORK PLAN

This work plan is a fist proposal and it could be changed. Anyway, it would be great to follow it to the end but perhaps some tasks could take longer hence I have left empty the last two weeks before the real deadline (September 30th)

TASK	DETAILS	DEADLINE
Study methods	Write down how they work	July 14th
Set up test environment	Create web graph at random	July 20th
Implement method 1	Codify the algorithm and write down results	July 27th
Implement method 2	Codify the algorithm and write down results	August 10th
Implement method 3	Codify the algorithm and write down results	August 24th
Improvement / suggestions	Write down	September 7th
Final write up		September 14th

5. REFERENCES

[1] M. Marchiori. The Quest for Correct Information on the Web: Hyper Search Engines. University of Padova, 1997.

[2] J. Kleinberg. Authoritative sources in a hyperlinked environment. In *Proc. Ninth Ann. ACM-SIAM Symp. Discrete Algorithms*, pages 668-677, ACM Press, New York, 1998.

[3] L. Page, S. Brin, R. Motwani and T. Winograd. The PageRank Citation Ranking: Bringing Order to the Web. Stanford University, 1999.

[4] IBM Almaden Researcn Center. Clever Searching. http://www.almaden.ibm.com/projects/clever.shtml.

[5] Y. Chen, Q. Gan, T. Suel. Local Methods for Estimating PageRank Values. Polytechnic University Brooklyn, 2004.