How is it made?
Google Search Engine

Joseph Khoury

November 15, 2010

Abstract

If you are one of the millions of people around the planet who use a search engine on a daily basis, you must have wondered at one point: how does the engine classify and display the information you are looking for? What makes your "favorite" search engine different from other search engines in terms of the relevance of the information you are looking for?

With billions of search requests a day, it is no surprise that Google is the search engine of choice of web surfers around the globe. The Mathematics behind the Google search algorithm (PageRank) could, however, come as a complete surprise to you. The purpose of this note is to explain, in as much self contained content as possible, the mathematical reasoning behind the PageRank algorithm.

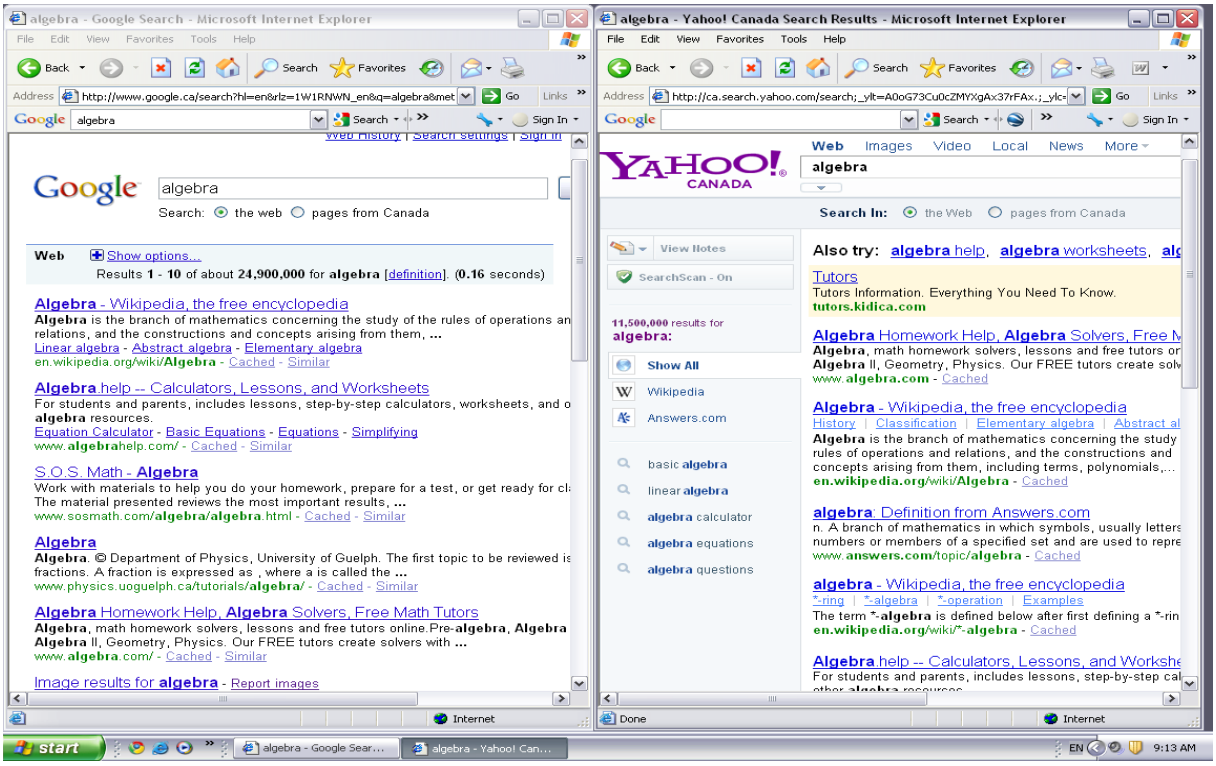
1 Introduction

The details of how exactly Google and other search engines look for information on the web and classify the importance of the pages containing the information you are looking for are certainly kept secret within a small circle of researchers and developers working for the company. However, at least for Google, the main component of its search mechanism has been known for a while as the *PageRank Algortithm*. The algorithm is named after Larry Page who, together with Sergey Brin, founded the company Google Inc in the late 90's. Here is how Google describes the PageRank algorithm on its cooperative site:

PageRank reflects our view of the importance of web pages by considering more than 500 million variables and 2 billion terms. Pages that we believe are important pages receive a higher PageRank and are more likely to appear at the top of the search results. PageRank also considers the importance of each page that casts a vote, as votes from some pages are considered to have greater value, thus giving the linked page greater value. We have always taken a pragmatic approach to help improve search quality and create useful products, and our technology uses the collective intelligence of the web to determine a page's importance.

The main tool used by PageRank is the theory of Markov chains, a well known process developed by the russian mathematician Andrei Markov at the beginning of the last century. In a nutshell, the process consists of a countable number of states and some given probabilities p_{ij}, with p_{ij} being the probability of moving from state j to state i.

Since algebra is the topic of the day, I decided to search the word "algebra" on both Google and Yahoo search Engines.

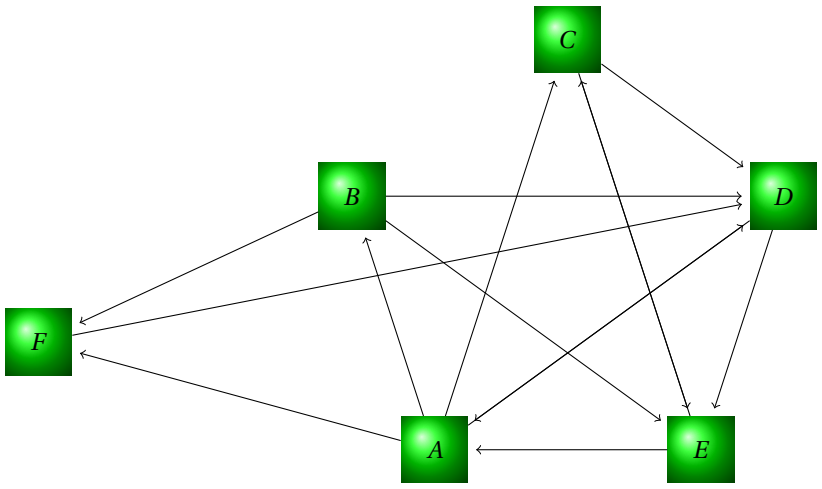


What Yahoo considers to be the top page for providing information about the search topic (Algebra Homework Help) ranked fifth in Google's ranking and the top page in Google's ranking (Wikipedia) came second in Yahoo's listing. Not very surprising considering how well known the word "Algebra" is. But what is more surprising is perhaps the fact that the page ranked fourth on Yahoo's list does not appear as one of the top 100 on Google's list. So who is to say what page contains more relevant information about "Algebra"? Is it only a matter of trust between the user and his or her favorite Search Engine? Google is so confident of its page ranking that it even added the "I am feeling lucky" button beside the search button that will take you right to the page ranked 1 on its listing.

It turns out that, at least from Google's perspective, a webpage is as important as the number of pages in the hyper world that point or link to it. But that is just one part of the story.

To start, let us represent every webpage with a **node** (or a **vertex**). If there is link from page "A" to page "B", then we make an arrow from node A to node B that we call an **edge**. The structure obtained is called a (directed) **graph**. To simplify our discussion, let us look at a hyperworld containing only six pages with the following links:

Example 1.1.



Note that a "double arrow" between two nodes means that there is a link from each of the two pages to the other. For example, there a link from page C to page E and vice versa in the above network.

The PageRank algorithm is based on the behavior of a *random web surfer* that we will refer to as *Joe* for simplicity. To surf the web, Joe would start at a page of his choice, then he randomly chooses a link from that page to another page and would continue this process until arriving at a page with no exterior links to other pages or until he (suddenly) decides to move to another page by other means than following a link from a current page, like for instance entering the URL of a page in the address bar. Two important aspects govern the behavior of Joe while surfing the web:

1. The choice of what page to visit next depends only on what page is Joe on now and not on the pages he previously visited;

2. Joe is resilient, in the sense that he will never give up on moving from one page to another either by following the link structure of the web or by other means.

Assume that there are n webpages in total ($n \approx 110000000$ in 2009), the PageRank algorithm creates a square $n \times n$ matrix H, that we call the *hyper matrix*, as follows:

- We assume that the webpages are given a certain numerical order $1, 2, 3, \ldots, n$, not necessarily by the order of importance. We just "label" the webpages using the integers $1, 2, 3, \ldots, n$.

- For any $i, j \in \{1, 2, \ldots, n\}$, the entry h_{ij} on the ith row and jth column of H represents the **probability** that Joe would go from page j to page i in one move (or one click). In other words, if the webpage j has a total of k_j links to others pages then

$$h_{ij} = \begin{cases} \frac{1}{k_j} & \text{if there is a link from page } j \text{ to page } i \\ 0 & \text{if there is no link from page } j \text{ to page } i \end{cases}$$

For the above model of Example 1.1, if the pages are ordered as follows $A = 1$, $B = 2$, $C = 3$, $D = 4$, $E = 5$ and $F = 6$, the hyper matrix is

$$H = \begin{bmatrix} 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{4} & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{4} & 0 & 0 & 0 & \frac{1}{2} & 0 \\ \frac{1}{4} & \frac{1}{3} & \frac{1}{2} & 0 & 0 & 1 \\ 0 & \frac{1}{3} & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ \frac{1}{4} & \frac{1}{3} & 0 & 0 & 0 & 0 \end{bmatrix}.$$

If we assume that there is an average of 10 links per page on the web, then the web hyper matrix is extremely "sparse". That is, there is an average of only 10 nonzero entries in each column of the matrix and the rest (billions of entries) are all zeros.

A page that does not link to any other page in the Network is called a *dangling node*. The presence of a dangling node in a network creates a column of zeros in the corresponding hyper matrix since if Joe lands on a dangling page, the probability that he leaves the page via a link is zero. Note that the Network in Example (1.1) above has no dangling nodes, we will deal with the dangling nodes problem a bit later in the discussion.

At any stage of the process, the notation $p_i(X)$ represents the probability that Joe lands on page X after i steps (or i clicks). If the page X is labeled using the integer j, then $p_i(X)$ is denoted by p_{ij}. The vector $p_i := \begin{bmatrix} p_{i1} & p_{i2} & \ldots & p_{in} \end{bmatrix}^t$ (where A^t means the transpose of the matrix A) is called the i**th probability distribution vector**. We also define the **initial probability vector** as being the vector with all 0's except for one entry equal to 1. That entry corresponds to the page were Joe initially starts his search. In Example (1.1), if Joe starts his search at the page C, then the initial probability distribution vector is $\begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}^t$.

Note that the ith column of the hypermatrix H is nothing but the initial probability vector corresponding to a start at the page labeled i.

At this point, the following questions become relevant:

1. Can we determine the probability distribution vector after k steps (or k clicks)? In other words, can we determine the probability of Joe being on page i of the web after k clicks?

2. Can we "predict" the behavior of Joe in the long run? That is, can we after a very big number of clicks determine the probability of Joe being on page i for any $i \in \{1,2,\ldots,n\}$?

3. If such a long term behavior of Joe can be determined, does it depend on the initial probability vector? That is, does it matter which page Joe starts his surfing with?

After certain refinements of the hypermatrix H, one can give definitive answers to all these questions.

Let us first look at some of these questions from the perspective of the 6-pages Network of Example 1.1. Assuming Joe starts at the page A, the initial distribution vector is $p_0 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^t$. After the first click, there is an equal probability of $\frac{1}{4}$ that Joe lands on either one of the pages B, C, D or F since these are the pages A links to. On the other hand, there is a zero probability that he lands on page E (again by using links). This means that after the first click, the probability distribution vector is $p_1 = \begin{bmatrix} 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & 0 & \frac{1}{4} \end{bmatrix}^t$. But note that

$$Hp_0 = \begin{bmatrix} 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{4} & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{4} & 0 & 0 & 0 & \frac{1}{2} & 0 \\ \frac{1}{4} & \frac{1}{3} & \frac{1}{2} & 0 & 0 & 1 \\ 0 & \frac{1}{3} & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ \frac{1}{4} & \frac{1}{3} & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{1}{4} \\ \frac{1}{4} \\ \frac{1}{4} \\ 0 \\ \frac{1}{4} \end{bmatrix} = p_1$$

Similarly, if Joe's initial distribution vector is $p_0 = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}^t$ (Joe starts at the page D), then after the first click there is an equal probability of $\frac{1}{2}$ to land on either one of the two pages A or E since these are the pages D links to and zero probability that he lands on any of the pages B, C, D and F by means of links. This suggests that after the first click, the probability distribution vector of Joe is $p_1 = \begin{bmatrix} \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 \end{bmatrix}^t$, and again

$$Hp_0 = \begin{bmatrix} 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{4} & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{4} & 0 & 0 & 0 & \frac{1}{2} & 0 \\ \frac{1}{4} & \frac{1}{3} & \frac{1}{2} & 0 & 0 & 1 \\ 0 & \frac{1}{3} & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ \frac{1}{4} & \frac{1}{3} & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{2} \\ 0 \\ 0 \\ 0 \\ \frac{1}{2} \\ 0 \end{bmatrix} = p_1.$$

This is hardly a coincidence. Suppose that the (only) entry 1 of the initial probability distribution vector is at the ith component (Joe starts at page i), then it is easy to see that Hp_0 is nothing but the ith column

of H which in turn is the probability distribution vector after the first click.

If p_k is the probability distribution vector after the kth click ($k \geq 1$), should we expect that $p_k = Hp_{k-1}$? Let us see what happens after the second click. Assume that Joe starts at the page A, then after the first click he is at one of the pages B, C, D or F with equal probability $\frac{1}{4}$. What is the probability of Joe landing on each of the pages after the second click? The answer depends on the paths available for Joe in his surf.

- The only way Joe can return to page A after the second click is that he follows the path $A \to D \to A$. This path happens with a probability of $\frac{1}{4}.\frac{1}{2}$ since once Joe is on page D he has two choices, page A or page E. So, $p_2(A) = \frac{1}{8}$ after the second click.
- Note that the only way to land on page B is from page A, meaning that there is no chance on landing on page B after the second click, $p_2(B) = 0$.
- Since the only pages linking to C are A and E and since there is no link from A to E, the chance of landing on C after the second click is zero, $p_2(C) = 0$.
- Landing on page D after the second click can de done through one of the following paths: $A \to C \to D$ with probability $\frac{1}{4}.\frac{1}{2} = \frac{1}{8}$ or $A \to C \to D$ with probability $\frac{1}{4}.\frac{1}{2} = \frac{1}{8}$ or $A \to F \to D$ with probability $\frac{1}{4}.1 = \frac{1}{4}$. So, $p_2(D) = \frac{1}{8} + \frac{1}{12} + \frac{1}{4} = \frac{11}{24}$.
- For the page E, Joe can reach it after the second click by following one of the following paths: $A \to B \to E$ with probability $\frac{1}{4}.\frac{1}{3} = \frac{1}{12}$, $A \to C \to E$ with probability $\frac{1}{4}.\frac{1}{2} = \frac{1}{8}$ or $A \to D \to E$ with probability $\frac{1}{4}.\frac{1}{2} = \frac{1}{8}$. So, $p_2(E) = \frac{1}{12} + \frac{1}{8} + \frac{1}{8} = \frac{1}{3}$.
- For page F, the only possible path is $A \to B \to F$ with probability $p_2(F) = \frac{1}{4}.\frac{1}{3} = \frac{1}{12}$

Note that the network has no dangling pages, so Joe must land on one of the pages after the second click. We should then expect that $p_2(A) + p_2(B) + p_2(C) + p_2(D) + p_2(E) + p_2(F) = 1$:

$$\frac{1}{8} + 0 + 0 + \frac{11}{24} + \frac{1}{3} + \frac{1}{12} = 1.$$

The probability distribution vector after the second click is then $p_2 = \begin{bmatrix} \frac{1}{8} & 0 & 0 & \frac{11}{24} & \frac{1}{3} & \frac{1}{12} \end{bmatrix}^t$. One must interpret the components of this vector as follows: starting at the page A and after the second click, Joe will land on page A with a probability of $\frac{1}{8}$, on page D with a probability of $\frac{11}{24}$, on page E with a probability of $\frac{1}{3}$, on page F with a probability of $\frac{1}{12}$ and there is no chance on landing on either one of pages B and C after the second click.

A closer look at the components of this new distribution vector p_2 reveals that they are obtained the following way:

$$p_2(A) = \frac{1}{8} = 0.0 + 0.\frac{1}{4} + 0.\frac{1}{4} + \frac{1}{2}.\frac{1}{4} + \frac{1}{2}.0 + 0.\frac{1}{4}$$

which is exactly the product of the first row of the hypermatrix H with the previous distribution column p_1. Similarly, $p_2(B)$ is the same as the product of the second row of H with p_1. Similar conclusions can be drawn for the other probability values. In other words,

$$p_2 = Hp_1 = H^2 p_0. \tag{1.0.1}$$

Continuing to the third click and beyond, one can now see that equation (1.0.1) can be generalized to

$$p_{k+1} = Hp_k = H^k p_0 \tag{1.0.2}$$

for any $k \geq 0$. The first 20 probability distribution vectors for example (1.1) above are given below (with components in decimal forms).

$$p_0 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad p_1 = \begin{bmatrix} 0.5 \\ 0 \\ 0 \\ 0 \\ 0.5 \\ 0 \end{bmatrix}, \quad p_2 = \begin{bmatrix} 0.125 \\ 0 \\ 0 \\ 0.4583 \\ 0.3333 \\ 0.0833 \end{bmatrix}, \quad \cdots, \quad p_{18} = \begin{bmatrix} 0.2319 \\ 0.0579 \\ 0.1670 \\ 0.2392 \\ 0.2239 \\ 0.0772 \end{bmatrix}, \quad p_{19} = \begin{bmatrix} 0.2315 \\ 0.0580 \\ 0.1699 \\ 0.2394 \\ 0.2239 \\ 0.0773 \end{bmatrix}, \quad p_{20} = \begin{bmatrix} 0.2317 \\ 0.0579 \\ 0.1698 \\ 0.2394 \\ 0.2240 \\ 0.0772 \end{bmatrix}$$

There is a clear indication that in the long run, Joe's probability distribution vector will be close to the vector

$$\pi = \begin{bmatrix} 0.231 \\ 0.058 \\ 0.169 \\ 0.239 \\ 0.224 \\ 0.078 \end{bmatrix}$$

In practical terms, this means that eventually Joe would visit page A with a probability of 23.1%, page B with a probability of 5.8% and so on. The page with the highest chance to be visited is clearly D with a probability of almost 24%. Note that the sum of the components of π is 1, making it a probability distribution vector. One can then "rank" the pages in Example (1.1) according to their chances of being visited after a sufficiently large walk of Joe on the Network: D, A, E, C, F, B would then be the order in which these pages would appear. The ranking vector π is called the **stationary distribution vector**.

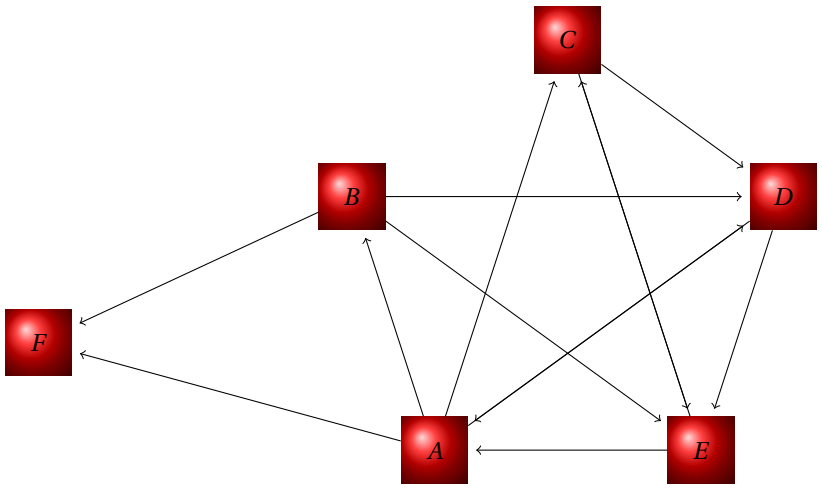
1.1 The dangling page problem

Example (1.1) seems to suggest that one can always rank the pages in any network just by making a sufficiently large walk to estimate the long term behavior of the probability distribution vector. Nothing could be further from the truth; things can get quickly out of hand if we consider a Network with dangling nodes

or a network with a trapping loop.

Let us slightly change the network in Example (1.1):

Example 1.2.



making F a dangling node. The hypermatrix of this new Network would be:

$$H = \begin{bmatrix} 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{4} & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{4} & 0 & 0 & 0 & \frac{1}{2} & 0 \\ \frac{1}{4} & \frac{1}{3} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{1}{3} & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ \frac{1}{4} & \frac{1}{3} & 0 & 0 & 0 & 0 \end{bmatrix}$$

Note that the last column is, as expected, the zero column due to the fact that page F does not link to any other page in the network. Starting at the page A and proceeding exactly the same way as in Example (1.1), the first 40 probability distribution vectors for Example (1.2) are (in decimal form):

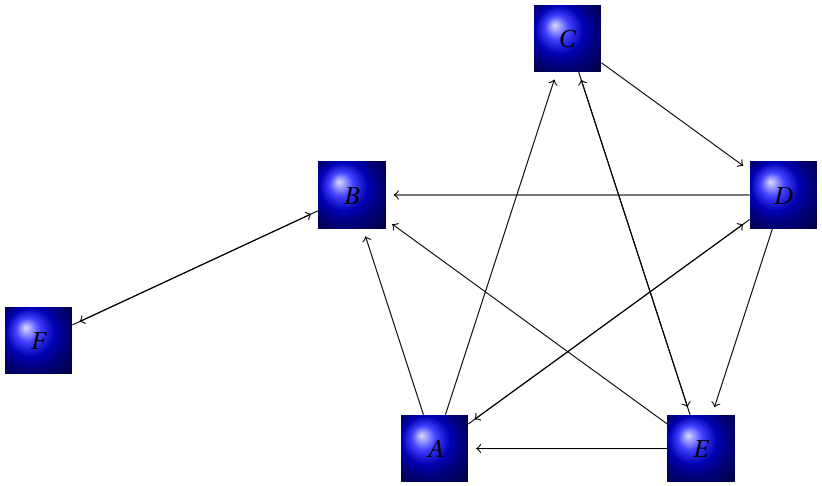
$$p_0 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad p_1 = \begin{bmatrix} 0 \\ 0.25 \\ 0.25 \\ 0.25 \\ 0 \\ 0.25 \end{bmatrix}, \quad p_2 = \begin{bmatrix} 0.125 \\ 0 \\ 0 \\ 0.2083 \\ 0.3333 \\ 0.0833 \end{bmatrix}, \quad \cdots, \quad p_{38} = \begin{bmatrix} 0.0065 \\ 0.0018 \\ 0.0054 \\ 0.0054 \\ 0.0065 \\ 0.0024 \end{bmatrix}, \quad p_{39} = \begin{bmatrix} 0.0060 \\ 0.0016 \\ 0.0050 \\ 0.0050 \\ 0.0060 \\ 0.0022 \end{bmatrix}, \quad p_{40} = \begin{bmatrix} 0.0054 \\ 0.0015 \\ 0.0045 \\ 0.0045 \\ 0.0054 \\ 0.0020 \end{bmatrix}$$

which seems to suggest that in the long run, Hp_k is approaching the zero vector. The above "ranking" procedure would not make much sense in this case.

1.2 The trapping loop problem

Another problem Joe could face following the link structure of the web is the chance that he could be trapped in a loop. Let us once more modify the Network of Example (1.1):

Example 1.3.



In this new Network, if Joe happens to land on page B then the only path he could take is the loop

$$B \to F \to B \to F \to B \to \dots$$

This suggests that the long term behavior of Joe can be described by the probability distribution vector $[\ 0 \quad \frac{1}{2} \quad 0 \quad 0 \quad 0 \quad \frac{1}{2}\ ]^t$. In terms of page ranking, this means that pages B and F will "absorb" the importance of all other pages in the network and that, of course, is not a reasonable ranking.

1.3 A possible Fix

In the light of the two complications Joe could face (dangling pages and trapping loops), one can reformulate the above three questions we posed earlier using a more mathematical language.

Given a square $n \times n$ matrix A and a vector $p_0 \in \mathbb{R}^n$:

1. Does the sequence of vectors p_0, $p_1 = Ap_0$, $p_2 = Ap_1$, ... (and in general $p_{j+1} = Ap_j$) always "converge" to a vector π?
2. If a vector π exists,
   (a) Is it unique?
   (b) Does it depend on the initial vector p_0?

For networks like the one in Example (1.1), with no dangling pages or trapping loops, it seems that the answers to allof these questions is yes. The sequence of Joe's probability distribution vectors

$$p_0, p_1, p_2, \ldots, p_k, \ldots$$

converges to a probability vector π (the long term probability distribution vector) that could be interpreted as a "ranking" of the pages (nodes) in the Network. If the ith component of π is the largest, then page i is ranked first, and so on.

In reality, a considerable percentage of actual webpages on the *World Wide Web* are indeed dangling pages, either because they are simply pictures, pdf files, postscript files, Excel or words files and similar forms or because at the time of the search Google data base was not updated. This makes the www hyperlink matrix a really *sparse* matrix, i.e a matrix with mostly zero entries. The above described algorithm of surfing the web based on outgoing links seems to be unreal unless the above problems are addressed.

After landing on a dangling page, the probability that Joe leaves the page to another via a link is zero, but he can still continue searching the web by other means, like for instance entering the Uniform Resource Locator (URL) directly into the web browser address bar. The following are two possible solutions for the dangling pages problem.

- One can assume that Joe leaves a dangling page with an equal probability of $\frac{1}{n}$ to visit any other page (by means other than following links). Consider the **dangling vector** d which is the row vector with the component d_i equals to 1 if page i is dangling and 0 otherwise. For example, the dangling vector in Example 1.2 above is $\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}^t$.

  Form the "new hypermatrix"

  $$S = H + \frac{1}{n}\mathbb{1}.d,$$

  where, as before, $\mathbb{1}$ is the column vector of $\mathbb{R}^n$ of all 1s. Simply put, the matrix S is obtained from H by replacing every zero column in the original hypermatrix H with the column $\begin{bmatrix} \frac{1}{n} & \frac{1}{n} & \cdots & \frac{1}{n} \end{bmatrix}^t$. The new matrix S is now *stochastic* (every column adds up to 1).

In Example (1.2), the new hypermatrix matrix is

$$
\begin{aligned}
S &= \begin{bmatrix} 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{4} & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{4} & 0 & 0 & 0 & \frac{1}{2} & 0 \\ \frac{1}{4} & \frac{1}{3} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{1}{3} & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ \frac{1}{4} & \frac{1}{3} & 0 & 0 & 0 & 0 \end{bmatrix} + \frac{1}{6} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{4} & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{4} & 0 & 0 & 0 & \frac{1}{2} & 0 \\ \frac{1}{4} & \frac{1}{3} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{1}{3} & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ \frac{1}{4} & \frac{1}{3} & 0 & 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & \frac{1}{6} \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{6} \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{6} \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{6} \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{6} \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{6} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & \frac{1}{6} \\ \frac{1}{4} & 0 & 0 & 0 & 0 & \frac{1}{6} \\ \frac{1}{4} & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{6} \\ \frac{1}{4} & \frac{1}{3} & \frac{1}{2} & 0 & 0 & \frac{1}{6} \\ 0 & \frac{1}{3} & \frac{1}{2} & \frac{1}{2} & 0 & \frac{1}{6} \\ \frac{1}{4} & \frac{1}{3} & 0 & 0 & 0 & \frac{1}{6} \end{bmatrix}
\end{aligned}
$$

- Another way to deal with dangling pages is to start by removing them, together with all the links leading to them, from the web. This will create a new well behaved (stochastic) hypermatrix that will hopefully have a stationary distribution vector π. We then use π as a ranking vector for the pages of the "new web". After this initial ranking is done, a dangling page X "inherits" the ranking from pages linking to it as follows. If page k is one of the pages linking to X with a total of m_k links to X and if r_k is the rank of page k, then we assign the sum

$$\sum_k \frac{r_k}{m_k}$$

  as the rank of the dangling page X (where k in the above sum runs over all the pages linking to X). In this fashion, the rankings of pages linking to a dangling page are in a way transferred to the dangling page.

Although the new matrix S is stochastic, there is still no guarantee that it will have a stationary probability distribution vector that could be used as a ranking vector (see Definition (1.1) and Theorem (2.1)). The inventors of PageRank (Page and Brin) made another adjustment to this end. While it is generally the case that web surfers follow the link structure of the web, an actual surfer might decide from time to time to "teleport" to a new page by entering a new destination in the address bar. From the new destination, the surfer continues to follow the links until he decides once more to teleport to a new page. To capture the surfer's mood, Page and Brin introduced a new matrix, called the **Google matrix**, as follows.

$$G = \alpha S + (1-\alpha)\frac{1}{n}\mathbb{1}.\mathbb{1}^t$$

where $\frac{1}{n}\mathbb{1}.\mathbb{1}^t$ is, of course, the $n \times n$ matrix where each entry is equal to $\frac{1}{n}$ representing the uniformly probable web teleporting process. α is a number between 0 and 1 called the "damping factor" representing

the proportion of times Joe teleports to a web page versus following the link the structure. For example, if $\alpha = 0.8$, then this would mean that 80% of the time Joe is following the link structure and 20% he teleports to a randomly chosen page. Note that if $\alpha = 0$, then $G = \frac{1}{n}\mathbb{1}.\mathbb{1}^t$ which means that Joe is teleporting all the time he is on the web. On the other extreme, if $\alpha = 1$, then $G = S$ which means that Joe is always following the link structure of the web. Realistically, α should then be *strictly* between 0 and 1 and more close to 1 than it is to 0 since Joe will more likely follow the links on the web. In the original article describing the PageRank algorithm ([1]), the authors used a damping factor of $\alpha = 0.85$.

Example 1.4. With $\alpha = 0.85$, the Google matrix for the Network in Example (1.1) (with entries rounded to 4 decimal places) is given by

$$G = \begin{bmatrix} 0.2500 & 0.2500 & 0.2500 & 0.4500 & 0.4500 & 0.1667 \\ 0.2375 & 0.2500 & 0.2500 & 0.2500 & 0.2500 & 0.1667 \\ 0.2375 & 0.2500 & 0.2500 & 0.2500 & 0.4500 & 0.1667 \\ 0.2375 & 0.3083 & 0.4500 & 0.2500 & 0.2500 & 0.1667 \\ 0.2500 & 0.3083 & 0.4500 & 0.4500 & 0.2500 & 0.1667 \\ 0.2375 & 0.3083 & 0.2500 & 0.2500 & 0.2500 & 0.1667 \end{bmatrix}.$$

For a general web, the Google matrix G satisfies the following properties:

- G is stochastic. In fact, write $S = [s_{ij}]$, then the sum of entries on the jth column of G is given by

$$\sum_{i=1}^{n}\left(\alpha s_{ij} + (1-\alpha)\frac{1}{n}\right) = \alpha \underbrace{\sum_{i=1}^{n} s_{ij}}_{=1} + n(1-\alpha)\frac{1}{n} = \alpha + (1-\alpha) = 1.$$

  If j is a dangling page, then $S_j = \frac{1}{n}[1, 1\ldots, 1]^t$ and the sum of the entries of the jth column of G is in this case equal to $\alpha\frac{1}{n}[1, 1\ldots, 1]^t + (1-\alpha)\frac{1}{n}[1, 1\ldots, 1]^t = 1$. If j is not a dangling page, then $S_j = [h_{1j}, h_{2j}\ldots, h_{nj}]^t$ (with $\sum_{i=1}^{n} h_{ij} = 1$) and the sum of the entries on the jth column of G is in this case equal to

$$\begin{aligned} \alpha\sum_{i=1}^{n} h_{ij} + (1-\alpha)\frac{1}{n}\sum_{i=1}^{n} 1 \quad &= \\ \alpha + (1-\alpha)\frac{1}{n}n = 1. & \end{aligned}$$

- G is positive. In fact, by the observation made above, the damping factor satisfies $0 < \alpha < 1$ (strict inequalities). Write $G = [g_{ij}]$, then $g_{ij} = \alpha s_{ij} + (1-\alpha)\frac{1}{n}$ where s_{ij} is either 0 or $\frac{1}{k_{ij}}$ for some positive integer $k_{ij} \leq n$. If $s_{ij} = 0$, then $g_{ij} = (1-\alpha)\frac{1}{n} > 0$. If $s_{ij} = \frac{1}{k_{ij}}$ for some positive integer $k_{ij} \leq n$, then

$$g_{ij} = \alpha s_{ij} + (1-\alpha)\frac{1}{n} = \alpha\frac{1}{k_{ij}} + (1-\alpha)\frac{1}{n} \geq \alpha\frac{1}{n} + (1-\alpha)\frac{1}{n} = \frac{1}{n} > 0.$$

  All entries of G are then positive.

In view of Theorem 2.1 below, the Google matrix G satisfies the desired requirements and will have a stationary probability distribution vector. We are now ready to define the Google page ranking.

Definition 1.1. Let $\pi = [\ \pi_1 \quad \pi_2 \quad \dots \quad \pi_n\ ]^t$ be the stationary probability distribution vector of the Google matrix G. The **Google rank** of page i is defined to be ith component π_i of the vector π. Page i comes before page j in Google ranking if and only if $\pi_i > \pi_j$.

From a search engine marketer's point of view, this means there are two ways in which PageRank can affect the position of your page on Google.

2 The Mathematics of PageRank

We begin this section with a quick review of some basic notions necessary for a full understanding of the mathematics behind Google's search algorithm. Linear Algebra is the main tool used in this algorithm and topics from this discipline are the main focus of review. The reader is assumed to be familiar with basic Matrix Algebra operations, like matrix addition, multiplication, inverse, determinant and algorithms of solving linear systems. Also assumed are the notions of subspaces and bases and dimensions of subspaces of $\mathbb{R}^n$.

Throughout, A denotes an $n \times n$ square matrix. The transpose of A is denoted by A^t.

2.1 The "Eigenstuff"

Definition 2.1. A *nonzero* vector X of $\mathbb{R}^n$ is called an **eigenvector** of A if there exists a scalar λ such that

$$AX = \lambda X. \tag{2.1.1}$$

The scalar λ is called an **eigenvalue** of A corresponding to the eigenvector X.

Note that relation (2.1.1) can be written as $(A - \lambda I)X = 0$, where I is the $n \times n$ *identity* matrix (having 1 on the main diagonal and 0 everywhere else). This shows that X is a solution to the linear *homogeneous* system $(A - \lambda I)X = 0$. The fact that X is assumed to be a nonzero vector implies that the system $(A - \lambda I)X = 0$ has a *nontrivial* solution and consequently, the coefficient matrix $(A - \lambda I)$ is not invertible. Therefore, $\det(A - \lambda I) = 0$ where "det" stands for the determinant of the matrix. The expression $\det(A - \lambda I)$ is clearly a polynomial of degree n in the variable λ usually referred to as the **characteristic polynomial** of A.

This suggests the following steps to find the eigenvalues and eigenvectors of A:

1. To find the eigenvalues of A, one has to find the roots of the characteristic polynomial of A; i.e, to solve the equation $\det(A - \lambda I) = 0$, called the **characteristic equation** of A, for the variable λ. This

is a polynomial equation of degree n in the variable λ which has n roots (not necessary distinct and could be complex numbers)

2. The set E_λ of all eigenvectors corresponding to an eigenvalue λ of A, together with the zero vector, form a subspace of $\mathbb{R}^n$ called the **eigenspace** corresponding to the eigenvalue λ. One usually needs a *basis* of E_λ. To this end, we solve the homogeneous system $(A-\lambda I)X=0$. As the coefficient matrix $(A-\lambda I)$ is not invertible, one should expect infinitely many solutions. Writing the general solution of the system $(A-\lambda I)X=0$ gives a basis of E_λ.

Example 2.1. *Find the eigenvalues of the given matrix, and for each eigenvalue find a basis for the corresponding eigenspace.*

$$A=\begin{bmatrix} 2 & 2 & 1 \\ 1 & 3 & 1 \\ 1 & 2 & 2 \end{bmatrix}.$$

Solution.

Using the properties of the determinant, the characteristic polynomial of A is

$$\begin{vmatrix} 2-\lambda & 2 & 1 \\ 1 & 3-\lambda & 1 \\ 1 & 2 & 2-\lambda \end{vmatrix} = \begin{vmatrix} 2-\lambda & 2 & 1 \\ 1 & 3-\lambda & 1 \\ 0 & \lambda-1 & 1-\lambda \end{vmatrix} = (\lambda-1)\begin{vmatrix} 2-\lambda & 2 & 1 \\ 1 & 3-\lambda & 1 \\ 0 & 1 & -1 \end{vmatrix}$$

$$= (\lambda-1)\begin{vmatrix} 2-\lambda & 2 & 3 \\ 1 & 3-\lambda & 4-\lambda \\ 0 & 1 & 0 \end{vmatrix} = -(\lambda-1)\begin{vmatrix} 2-\lambda & 3 \\ 1 & 4-\lambda \end{vmatrix} =$$

$$-(\lambda-1)(\lambda^2-6\lambda+5) = -(\lambda-1)^2(\lambda-5)$$

The eigenvalues of A are then $\lambda_1=1$ of *algebraic multiplicity* 2 and $\lambda_1=5$ of *algebraic multiplicity* 1.

For the eigenspace corresponding to $\lambda_1=1$, we write the general solution of the homogeneous system $(A-\lambda_1 I)X=0$ (I being the 3×3 identity matrix):

$$\left|\begin{array}{ccc:c} 2-\lambda_1 & 2 & 1 & 0 \\ 1 & 3-\lambda_1 & 1 & 0 \\ 1 & 2 & 2-\lambda_1 & 0 \end{array}\right| = \left|\begin{array}{ccc:c} 1 & 2 & 1 & 0 \\ 1 & 2 & 1 & 0 \\ 1 & 2 & 1 & 0 \end{array}\right| \sim \left|\begin{array}{ccc:c} 1 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right|$$

The two variables x_2 and x_3 are free variables, and $x_1=-2x_2-x_3$. So the general solution of the homogeneous system $(A-\lambda_1 I)X=0$ is:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -2x_2-x_3 \\ x_2 \\ x_3 \end{bmatrix} = x_2\begin{bmatrix} -2 \\ 1 \\ 0 \end{bmatrix} + x_3\begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} = x_2 v_1 + x_3 v_2.$$

So $E_{\lambda_1} = \text{span}\{v_1,\ v_2\}$. Since v_1 and v_2 are linearly independent, they form a basis of the eigenspace E_{λ_1}.

For the eigenspace corresponding to $\lambda_2 = 5$, we write the general solution of the homogeneous system $(A - \lambda_2 I)X = 0$:

$$\left|\begin{array}{ccc:c} 2-\lambda_2 & 2 & 1 & 0 \\ 1 & 3-\lambda_2 & 1 & 0 \\ 1 & 2 & 2-\lambda_2 & 0 \end{array}\right| = \left|\begin{array}{ccc:c} -3 & 2 & 1 & 0 \\ 1 & -2 & 1 & 0 \\ 1 & 2 & -3 & 0 \end{array}\right| \sim \left|\begin{array}{ccc:c} 1 & -2 & 1 & 0 \\ -3 & 2 & 1 & 0 \\ 1 & 2 & -3 & 0 \end{array}\right| \sim \left|\begin{array}{ccc:c} 1 & -2 & 1 & 0 \\ 0 & -4 & 4 & 0 \\ 0 & 4 & -4 & 0 \end{array}\right|$$

$$\sim \left|\begin{array}{ccc:c} 1 & -2 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right| \sim \left|\begin{array}{ccc:c} 1 & 0 & 3 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right|$$

Only x_3 is a free variable. Moreover, $x_1 = -3x_3$ and $x_2 = -x_3$. This shows that the eigenspace E_{λ_2} is one-dimensional with the vector $\begin{bmatrix} -3 & -1 & 1 \end{bmatrix}^t$ as a basis.

Lemma 2.1. *(1) The characteristic polynomials of A and A^t are equal. In particular, a square matrix has the same eigenvalues as its transpose;*
(2) If v is an eigenvector of A corresponding to the eigenvalue λ, then for any non-negative integer k, v is an eigenvector of A^k corresponding to the eigenvalue λ^k.

Proof
(1) The proof relies on the fact that the determinant of a square matrix is equal to the determinant of its transpose. If $c_A(\lambda)$ is the characteristic polynomial of A, then

$$c_A(\lambda) = det(A - \lambda I) = det(A - \lambda I)^t = det(A^t - \lambda I^t) = det(A^t - \lambda I) = c_{A^t}(\lambda).$$

The second statement follows directly from the definition of an eigenvalue.
(2) Let v be an eigenvector corresponding to λ. Then $Av = \lambda v$. If $k > 0$, then

$$A^k v = A^{k-1}(Av) = A^{k-1}(\lambda v) = A^{k-2}(\lambda Av) = A^{k-2}(\lambda^2 v) = \cdots = A^0(\lambda^k v) = I(\lambda^k v) = \lambda^k v.$$

This shows that λ^k is an eigenvalue of A^k corresponding to the eigenvector v. ■

2.2 Stochastic matrices

Google PageRank Algorithm uses a special "probabilistic approach" to rank the importance of pages on the web. The probability of what page a virtual surfer chooses to visit next depends solely on the current page the surfer is on and not on pages he previously visited. The matrices arising from such an approach are called stochastic.

Definition 2.2. The square matrix $A = [a_{ij}]$ is called **stochastic** if each of its entries is a non-negative real number and the entries on each column add up to 1. In other words

$$\forall i, j,\ a_{ij} \geq 0 \text{ and for each } k,\ \sum_{s=1}^{n} a_{sk} = 1.$$

Example 2.2. The matrices

$$\begin{bmatrix} \frac{1}{2} & 0 \\ \frac{1}{2} & 1 \end{bmatrix}, \quad \begin{bmatrix} \frac{1}{2} & 0 & \frac{1}{3} \\ 0 & \frac{2}{3} & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{3} \end{bmatrix}, \quad \begin{bmatrix} \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & 0 \\ 0 & 0 & \frac{1}{4} & 0 \\ 0 & \frac{2}{3} & \frac{1}{4} & 1 \\ \frac{1}{2} & 0 & \frac{1}{4} & 0 \end{bmatrix}$$

are examples of stochastic matrices.

Definition 2.3. We say that the matrix $A = [a_{ij}]$ is **positive**, and we write $A > 0$, if $a_{ij} > 0$ for all $1 \leq i, j \leq n$. We say that A is **non-negative**, and we write $A \geq 0$ if $a_{ij} \geq 0$ for all $1 \leq i, j \leq n$. The matrix A is called **regular** if A^k is positive for some $k \geq 1$.

Example 2.3. The matrix $\begin{bmatrix} 1 & 2 \\ 5 & 3 \end{bmatrix}$ is positive while $\begin{bmatrix} 1 & 2 \\ 5 & 0 \end{bmatrix}$ is not. However, $\begin{bmatrix} 1 & 2 \\ 5 & 0 \end{bmatrix}^2 = \begin{bmatrix} 11 & 2 \\ 5 & 10 \end{bmatrix}$ is positive, so $\begin{bmatrix} 1 & 2 \\ 5 & 0 \end{bmatrix}$ is regular.

Remark 2.1. Every positive matrix is in particular regular (just take $k = 1$). However, not all non-negative matrices are regular. For example, the matrix $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ is non-negative but not regular (Why?).

The first result we need follows almost from the definition of stochastic matrices.

Lemma 2.2. *(1) If A is a stochastic matrix and p is a column vector with non-negative components that add up to 1, then the same is true for the column vector Ap;*
(2) The product of two stochastic matrices is stochastic. In particular, if A is a stochastic matrix, then A^k is stochastic for any non-negative integer k.

Proof
For part (1), the sum of the components of the vector Ap is given by

$$\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}p_j = \sum_{j=1}^{n}\sum_{i=1}^{n} a_{ij}p_j = \sum_{j=1}^{n} p_j \underbrace{\sum_{i=1}^{n} a_{ij}}_{1} = \sum_{j=1}^{n} p_j = 1.$$

For part (2), let $A = [a_{ij}]$ and $B = [b_{ij}]$ be two stochastic matrices (of the same size $n \times n$). Note first that the components of AB are clearly non-negative since A and B consist of solely non-negative entries. The jth column of AB is Ab_j where b_j is the jth column of B. By part (1), the components of Ab_j add up to 1. Consequently, AB is stochastic. The second statement of (2) follows easily using a simple induction argument on the non-negative integer k. This finishes the proof of the lemma. ■

The next Proposition provides some special properties of stochastic matrices that are essential for the well functioning of the PageRank algorithm.

Proposition 2.1. *If $A = [a_{ij}]$ is a stochastic matrix, then the following hold.*

1. *$\lambda = 1$ is an eigenvalue for A;*
2. *If A is regular, then any eigenvector corresponding to the eigenvalue 1 of A has all positive or all negative components;*
3. *If λ is any eigenvalue of A, then $|\lambda| \leq 1$;*
4. *If A is regular, then for any eigenvalue λ of A other than 1 we have $|\lambda| < 1$.*

Proof

Consider the vector $\mathbb{1} = [\ 1 \quad 1 \quad \dots \quad 1\ ]$ of $\mathbb{R}^n$. The ith component of the vector $A^t\mathbb{1}^t$ is given by

$$\sum_{k=1}^{n} a_{ki}.1 = \sum_{k=1}^{n} a_{ki} = 1$$

since A is stochastic. This shows that $A^t\mathbb{1}^t = \mathbb{1}^t$ and so $\lambda = 1$ is an eigenvalue of A^t with $\mathbb{1}^t$ as corresponding eigenvector. Lemma 2.1 shows that $\lambda = 1$ is an eigenvalue for A. Part 1 of the Proposition is proved.

For part 2, we may assume that A is positive by the second part of Lemma 2.1. We use a proof by contradiction. Let $v = [\ v_1 \quad v_2 \quad \dots \quad v_n\ ]^t$ be an eigenvector of the eigenvalue 1 containing components of mixed signs. Since $Av = v$, we have that $v_i = \sum_{k=1}^{n} a_{ik}v_k$ and the terms $a_{ik}v_k$ in this sum are of mixed signs since $a_{ik} > 0$ for each k. Therefore,

$$|v_i| = \left|\sum_{k=1}^{n} a_{ik}v_k\right| < \sum_{k=1}^{n} a_{ik}|v_k| \tag{2.2.1}$$

by the triangular inequality. The strict inequality occurs because the terms $a_{ik}v_k$ in this sum are of mixed signs. Taking the sum from $i = 1$ to $i = n$ on both sides in (2.2.1) yields:

$$\sum_{i=1}^{n} |v_i| < \sum_{i=1}^{n}\sum_{k=1}^{n} a_{ik}|v_k| = \sum_{k=1}^{n} \underbrace{\left(\sum_{i=1}^{n} a_{ik}\right)}_{=1} |v_k| = \sum_{k=1}^{n} |v_k|.$$

This is clearly a contradiction. We conclude that the vector v cannot have both positive and negative components at the same time. Assume that $v_i \geq 0$ for all i, then for each i, the relation $v_i = \sum_{k=1}^{n} a_{ik}v_k$ together with the fact that $a_{ik} > 0$ imply that $v_i > 0$ since at least one of the v_k's is not zero (v is an eigenvector). Similarly, if $v_i \leq 0$ for all i then $v_i < 0$ for all i. This proves part 2 of the Proposition

For part (3), we use again the fact that A and A^t have the same eigenvalues. Let λ be any eigenvalue of A^t and let $v = [\ v_1 \quad v_2 \quad \dots \quad v_n\ ]^t \in \mathbb{R}^n$ be a corresponding eigenvector. Suppose that the component v_j of v satisfies $|v_j| = \max\{|v_i|;\ i = 1, \dots n\}$ so that for any $l = 1, 2, \dots, n$, $|v_l| \leq |v_j|$. By taking the absolute values of the jth components on both sides of $\lambda v = A^t v$, we get that $|\lambda v_j| = \left|\sum_{i=1}^{n} a_{ij}v_i\right|$. Therefore,

$$|\lambda v_j| = |\lambda||v_j| = \left|\sum_{i=1}^{n} a_{ij}v_i\right| \quad \leq \quad \sum_{i=1}^{n} a_{ij}|v_j| = |v_j|\sum_{i=1}^{n} a_{ij} = |v_j| \quad \left(\text{since } \sum_{i=1}^{n} a_{ij} = 1\right)$$

The inequality $|\lambda||v_j| \leq |v_j|$ implies that $|\lambda| \leq 1$ (remember that $|v_j| \neq 0$) and part (3) is proved.

For part 4, assume first that A (hence A^t) is a positive matrix. Let λ be an eigenvalue of A^t with $|\lambda| = 1$. We show that $\lambda = 1$. As in the proof of part 3, let $v = [\ v_1 \quad v_2 \quad \dots \quad v_n\ ]^t \in \mathbb{R}^n$ be a an eigenvector corresponding to λ with $|v_j| = \max\{|v_k|,\ k = 1,\dots,n\}$, then

$$|v_j| = 1.|v_j| = |\lambda||v_j| = |\lambda v_j| = \left|\sum_{i=1}^{n} a_{ij} v_i\right| \leq \sum_{i=1}^{n} a_{ij}|v_i| \leq \underbrace{\sum_{i=1}^{n} a_{ij}}_{=1}|v_j| = |v_j|. \tag{2.2.2}$$

This shows that the last two inequalities in (2.2.2) are indeed equal signs (bounded on the left and on the right by $|v_j|$). The first inequality is an equal sign if and only if all the terms in the sum $\sum_{i=1}^{n} a_{ij} v_i$ have the same sign (all positive or all negative) and hence all the v_i's are of the same sign (note that this gives another proof of part 2). The fact that the second inequality is indeed an equal sign gives

$$\sum_{i=1}^{n} a_{ij}(|v_j| - |v_i|) = 0. \tag{2.2.3}$$

But $a_{ij} > 0$ and $|v_j| - |v_i| \geq 0$ for all $i = 1,2,\dots,n$. Equation (2.2.3) implies that $|v_j| - |v_i| = 0$ for all $i = 1,2,\dots,n$. This, together with the fact that all the v_i's have the same sign, imply that the vector v is a scalar multiple of $\mathbb{1} = [\ 1 \quad 1 \quad \dots \quad 1\ ]^t$. This shows that the eigenspace of A^t corresponding to the eigenvalue λ is one dimensional equals to span$\{\mathbb{1}\}$. In particular, $\mathbb{1}$ is an eigenvector corresponding to λ and consequently, $A^t\mathbb{1} = \lambda\mathbb{1}$. But the vector $\mathbb{1}$ also satisfies $A^t\mathbb{1} = \mathbb{1}$ by the proof of part (1) of this Proposition. This shows that $\lambda\mathbb{1} = \mathbb{1}$ which forces λ to equal 1.

Assume next that A is regular and choose a positive integer k such that $A^k > 0$. Let λ be an eigenvalue of A satisfying $|\lambda| = 1$. Then part 2 of Lemma 2.1 shows that λ^k is an eigenvalue of A^k and λ^{k+1} is an eigenvalue of A^{k+1}. Since both A^k and A^{k+1} are positive matrices, we must have that $\lambda^k = \lambda^{k+1} = 1$ (by the proof of the positive case). This last relation can be rearranged as $\lambda^k(\lambda - 1) = 0$ which gives that $\lambda = 1$ since $\lambda^k \neq 0$ (remember we are assuming that $|\lambda| = 1$). ■

To prove the main Theorem behind PageRank algorithm, we still need a couple of basic results.

Lemma 2.3. *Let $n \geq 2$, u, v two linearly independent vectors in $\mathbb{R}^n$. Then, we can choose two scalars s and t not both zero at the same time such that the vector $w = su + tv$ has components of mixed signs.*

Proof

The fact that the vectors u, v are linearly independent implies that none of them is the zero vector. Let α be the sum of all the components of the vector u. If $\alpha = 0$, then u must contain components of mixed signs. The values s = 1 and t = 0 will do the trick in this case. If $\alpha \neq 0$, let $s = -\frac{\beta}{\alpha}$ where β is the sum of all the components of the vector v. For $t = 1$, the sum of the components of the vector $w = su + tv$ is zero. On the other hand, the vector w is nonzero since otherwise the vectors u and v would be linearly dependent. We conclude that the components of w are of mixed signs. ■

Proposition 2.2. *If A is a regular and stochastic matrix, then the eigenspace corresponding to the eigenvalue 1 of A is one-dimensional.*

Proof

Suppose not. Then we can choose two linearly independent eigenvectors u, v corresponding to the eigenvalue 1. By Lemma (2.3) above, we can choose two scalars s and t not both zero at the same time such that the vector $w = su + tv$ has components of mixed signs. The vector w is also an eigenvector corresponding to the eigenvalue 1 of A. That is a contradiction to part 2 of proposition (2.1). This shows that no two eigenvectors of the eigenvalue 1 can be linearly independent. Hence, the eigenspace corresponding to the eigenvalue 1 is one-dimensional. ■

We now can state and prove the main Theorem of this section.

Theorem 2.1. *If A is an $n \times n$ regular stochastic matrix, then there exists a **unique** vector $\pi = [\ \pi_1 \quad \pi_2 \quad \dots \quad \pi_n\ ]^t \in \mathbb{R}^n$ such that $A\pi = \pi$ and*

$$\sum_{i=1}^{n} \pi_i = 1, \ \textit{and} \ \pi_i > 0 \ \textit{for all} \ i = 1, \dots, n.$$

Proof

By Propositions 2.2 and 2.1 above, the eigenspace E_1 corresponding to the eigenvalue 1 of A can be written as $E_1 = \text{Span}\{v\}$ for some vector v with all positive or all negative components. Let $\pi = \frac{1}{a} v$ where a is the sum of all components of v. Then π is also an eigenvector of A corresponding to the eigenvalue 1 (hence $A\pi = \pi$) and it is the only one satisfying the required conditions. ■

2.3 An eigenvector for a $25000000000 \times 25000000000$ matrix, really?

In theory, the Google matrix has a stationary probability distribution vector π, which is an eigenvector corresponding to the eigenvalue 1 of the matrix. This should be, at least in theory, a straightforward task that can be done by any student who completed a first year university linear algebra course. But remember that we are dealing with an $n \times n$ matrix with n measured in billions and maybe in trillions by the time you read this work. Even the most powerful machines and computational algorithms we have in our days will have enormous difficulties computing π.

One of the oldest and simplest methods to compute numerically the eigenvector of a given square matrix is what is known in the literature as the **power method**. This method is simple, elementary and easy to implement in a computer algebra software, provided that the matrix has a dominant eigenvalue (that is an eigenvalue that is strictly larger in absolute value than any other eigenvalue of the matrix), but it is in general slow in giving a satisfactory estimation. However, considering the nature of the Google matrix G, the power method is well-suited to compute the stationary probability distribution vector. This computation was described by Cleve Moler, the founder of Matlab as "The World' s Largest Matrix Computation"

in an article published in Matlab newsletter in October 2002.

To explain the power method, we will assume for simplicity that the Google matrix G, in addition of being positive and stochastic, has n distinct eigenvalues, although this is not a necessary condition. This makes G a *diagonalizable* matrix and one can choose a basis $\{v_1,\ v_2,\ \ldots,\ v_n\}$ of $\mathbb{R}^n$ formed by eigenvectors of G (each v_i is an eigenvector of G). By Proposition 2.1 above, we know that $\lambda = 1$ is a **dominant eigenvalue** (part 4 of Proposition 2.1). Rearrange the eigenvectors $v_1,\ v_2,\ \ldots,\ v_n$ of G so that the corresponding eigenvalues decrease in absolute value:

$$1 > |\lambda_2| \geq |\lambda_3| \geq \ldots \geq |\lambda_n|$$

with the first inequality being strict. Note also that for each i, and for each positive integer k, we have

$$G^k v_i = G^{k-1}(Gv_i) = G^{k-1}(\lambda_i v_i) = \lambda_i G^{k-1} v_i = \ldots = {\lambda_i}^k v_i. \tag{2.3.1}$$

We can clearly assume that $v_1 = \pi$ is G's stationary probability distribution vector. Starting with any vector $p_0 \in \mathbb{R}^n$ with non-negative components that add to 1, we write p_0 in terms of the basis vectors:

$$p_0 = a_1\pi + a_2 v_2 + \cdots + a_n v_n, \tag{2.3.2}$$

where each a_i is a real number. Then we compute the vectors

$$p_1 = Gp_0,\ p_2 = Gp_1 = G^2 p_0,\ \ldots,\ p_k = Gp_{k-1} = G^k p_0,\ \ldots$$

Using the decomposition of p_0 given in 2.3.2 and relation 2.3.1 above we can write

$$\begin{aligned} p_k = G^k p_0 &= G^k[a_1\pi + a_2 v_2 + \cdots + a_n v_n] \\ &= a_1 1^k \pi + {\lambda_2}^k v_2 + \cdots + {\lambda_n}^k v_n \\ &= a_1\pi + {\lambda_2}^k v_2 + \cdots + {\lambda_n}^k v_n. \end{aligned}$$

By Lemma 2.2 above, the sum of the components of the vector $G^k p_0$ is 1. Taking the sum of components on each side of the equation $G^k p_0 = a_1\pi + {\lambda_2}^k v_2 + \cdots + {\lambda_n}^k v_n$ gives

$$1 = a_1 \sum_{j=1}^{n} \pi_j + \sum_{j=1}^{n}\sum_{i=2}^{n} a_j {\lambda_j}^k v_{ji} = a_1 + \sum_{j=1}^{n}\sum_{i=2}^{n} a_j {\lambda_j}^k v_{ji}. \tag{2.3.3}$$

Since $|\lambda_i| < 1$ for each $i = 2,\ldots,n$, $\lim_{k\to+\infty} {\lambda_i}^k = 0$ and so taking the limit as k approaches infinity on both sides of equation (2.3.3) gives that $a_1 = 1$. Therefore,

$$p_k = G^k p_0 = \pi + {\lambda_2}^k v_2 + \cdots + {\lambda_n}^k v_n. \tag{2.3.4}$$

Again, taking the limit as k approaches infinity gives that the sequence of vectors $p_0,\ Gp_0,\ G^2 p_0,\ \ldots,\ G^k p_0,\ \ldots$ converges to the stationary probability distribution vector π.

In theory, one can use the power method to estimate π. But how many iterations do we need to compute in order to get an acceptable approximation of π? In other words, what value of k should we choose in order for $G^k p_0$ to be "close enough" to π? The answer is in the magnitude of the second largest eigenvalue (in absolute value). To see this, denote by $\|v\|$ the "norm" of $v = [\ v_1 \quad v_2 \quad \dots \quad v_n\ ] \in \mathbb{R}^n$ in the following sense:

$$\|v\| = \sum_{i=1}^{n} |v_i|.$$

Scaling the vectors v_i's in the basis considered above by replacing each v_i with $\frac{1}{\|v_i\|} v_i$ (the vector π being already of norm 1) gives a new "normalized" (each vector is of norm 1) basis formed also by eigenvectors of G. We can then assume without loss of generality that $\|v_i\| = 1$ for all $i = 2,3,\dots,n$. Taking the norm on both sides of 2.3.4 gives

$$\begin{aligned}
\|p_k - \pi\| = \|{\lambda_2}^k v_2 + {\lambda_3}^k v_3 + \cdots + {\lambda_n}^k v_n\| &\leq |\lambda_2|^k \|v_2\| + |\lambda_3|^k \|v_3\| + \cdots + |\lambda_n|^k \|v_n\| \\
&= |\lambda_2|^k \left(\|v_2\| + \left|\frac{\lambda_3}{\lambda_2}\right|^k \|v_3\| + \cdots + \left|\frac{\lambda_n}{\lambda_2}\right|^k \|v_n\| \right) \\
&= |\lambda_2|^k \left(1 + \left|\frac{\lambda_3}{\lambda_2}\right|^k + \cdots + \left|\frac{\lambda_n}{\lambda_2}\right|^k \right)
\end{aligned}$$

As k approaches infinity, $\|p_k - \pi\| \leq |\lambda_2|^k$ since $\left|\frac{\lambda_i}{\lambda_2}\right| < 1$ for each $i = 3,4,\dots,n$. So, $|\lambda_2|^k$ serves as an upper bound on the error in estimating π using p_k, and so the smaller $|\lambda_2|$ is, the better this approximation and the quicker the convergence of the sequence is.

It was proven that for the Google matrix $G = \alpha S + (1-\alpha)\frac{1}{n}\mathbb{1}.\mathbb{1}^t$, $\lambda_2 = \alpha$ ([2]). This creates a bit of a dilemma since on one hand, one wants to make α closer to 1 than 0 to reflect the fact that Joe follows the link structure more often than teleporting on a new page, and on the other hand one would like to consider smaller values for α to accelerate the convergence of the iteration sequence given by $p_k = G^k p_0$ and get the estimate for the ranking vector π. The compromise was to take $\alpha = 0.85$. With this choice, Brin and Page reported that between 50 and 100 iterations are required to obtain a decent approximation to π. The calculation is reported to take a few days to complete.

Another particularity of the Google matrix G that makes the power method very practical in this case is the fact that its hypermatrix component is very sparse. Recall that $G = \alpha S + (1-\alpha)\frac{1}{n}\mathbb{1}.\mathbb{1}^t$, and $S = H + \frac{1}{n}\mathbb{1}.d$ as above, so

$$\begin{aligned}
Gp_k &= \left[\alpha\left(H + \frac{1}{n}\mathbb{1}.d\right) + (1-\alpha)\frac{1}{n}\mathbb{1}.\mathbb{1}^t\right].p_k \\
&= \alpha H.p_k + \frac{\alpha}{n}\mathbb{1}.d.p_k + \frac{(1-\alpha)}{n}\mathbb{1}.\mathbb{1}^t.p_k \\
&= \alpha H.p_k + \frac{\alpha}{n}\mathbb{1}.d.p_k + \frac{(1-\alpha)}{n}B.p_k
\end{aligned}$$

where $B = \mathbb{1}.\mathbb{1}^t$ is the constant matrix where each entry is 1. Since most of the entries in H are zeros, computing $H.p_k$ requires very little effort (on average, only ten entries per column of H are nonzero). The computations $\frac{\alpha}{n}\mathbb{1}.d.p_k$ and $\frac{(1-\alpha)}{n}B.p_k$ can be done by simply adding the current probabilities (components of p_k) to the dangling pages and all the web pages respectively.

2.4 Summary

First one has to understand that Google PageRank is only one ranking criteria Google uses. You can think of it as a multiplying factor in the global Google relevance algorithm. The higher this factor is, the more important the page is.

Unlike what most people think, the PageRank algorithm has absolutely nothing to do with the *relevance* of the search terms you enter in the Google bar. It is again only one aspect of the global Google ranking algorithm. Links leading to a page X and links out from from pages linking to X have the bigger effect.

Here are the basic steps in the period before and after you enter your query.

- Google is continuously crawling the web in real time with software called “Googlebots”. A Google crawler visits a page, copies the content and follows the links from that page to the pages linked to it, repeating this process over and over until it has crawled billions of pages on the web.
- After processing these pages and their contents, Google creates an index similar in its idea to a normal index you find at the end of a book.
- However, Google index is different from a regular index since not only topics are displayed but rather every single word a crawler has recorded together with their location on the pages and other information.
- Because of the size of Google index, it is divided into pieces and stored on thousands of machines around the globe.
- So every time you enter a query in Google search box, the query is sent to Google computers (depending on your geographic location).
- Google algorithm first calculates the relevance of pages containing the search words in its index creating a preliminary list.
- The "relevance" of each page on this preliminary list is then multiplied with the corresponding PageRank of the page to produce the final list on your screen (together with a short text summary for each result).

It is amazing what a little knowledge of Mathematics can produce.

References

[1] Lawrence Page, Sergey Brin, Rajeev Motwani, Terry Winograd, *The PageRank citation ranking: Bringing order to the Web*, Stanford Technical report, 1999.

[2] Taher Haveliwala, Sepandar Kamvar, *The second eigenvalue of the Google matrix*, Stanford Technical report, June, 2003.