HOW GOOGLE RANKS WEB PAGES

(Summary)

There are many factors that determine how Google ranks the search engine results. However, it is insisted by Google that the heart of its search engine software is the PageRank algorithm. A few quick searches on the Internet reveal that both the business and academic communities hold PageRank in high regard.
In the PageRank method, the ranking of pages is based solely on how pages are linked and not on the content of the pages or on how often the pages are visited. Since the web is constantly changing, the rankings change, too. In fact, Google calculates the ranks once a month.

The Mathematics behind the PageRank

PageRank is a link analysis algorithm that assigns a numerical weighting to each element of a hyperlinked set of documents, such as the World Wide Web, with the purpose of "measuring" its relative importance within the set. The algorithm models the behavior of an idealized random Web surfer. This Internet user randomly chooses a webpage to view from the listing of available webpages. Then, the surfer randomly selects a link from that webpage to another webpage. The surfer continues the process of selecting links at random from successive webpages until deciding to move to another web page by some means other than selecting a link. The choice of which web page to visit next does not depend on the previously visited webpages, and the idealized Web surfer never grows tired of visiting webpages. Thus, the PageRank score of a webpage represents the probability that a random Web surfer chooses to view the webpage. In order to obtain the final PageRank scores, where the rankings form the entries of a vector V$\in R^N$, and N is the total number of pages on the web, there are a number of steps that explain this algorithm. First of all, Let N be the total number of web pages and A be the N by N matrix whose ij entry is

$$\text{aij} = \begin{cases} 1 \text{ if page j has a link to page i} \\ 0 \text{ if not.} \end{cases}$$

Then vector v can be written as follow: $v = Au$
Where $u \in R^N$ is the vector whose components are 1.
Secondly, each directed link from web page j to web page i is normalized by the total numbers of links that come out from that web page j. In another way, each aij element is divided by the summation of elements in column j (nij), where $nij = \sum_i aij$. Nevertheless, to avoid the problem of division by zero if there is no link coming out from web page j, it is assumed that this page has a link to every page, including itself. Now, v can be represented more concisely as follow: $v = Pu$
Where P is the N by N matrix whose ij entry is

$$Pij = \frac{aij}{nj}$$

Then, since vj is supposed to reflect the value or importance of page j, each Pij is multiplied by vj , or in matrix notation:

$$v = Pv$$

Therefore, v should be an eigenvector of P with eigenvalue 1. Since P is a Markov matrix. Then $\lambda = 1$ is an eigenvalue of P. Furthermore, there is an eigenvector v with eigenvalue 1 and with each entry greater than or equal zero.
Finally, although the matrix P does have an eigenvector with eigenvalue, if there are several clusters of webpages not connected to each other, then P will have a set of two or more linearly independent eigenvectors with eigenvalue 1. In that case, the equation $v = Pv$ will not determine a unique ranking. Another problem with $v = Pv$ is that many pages are likely to be tied with a score of 0. Indeed, if it is possible by following links to get from page j to page i but not vice versa, then vj turns out to be 0. Google avoids these problems by replacing P by another Markov matrix Q.

$$Q = rP + (1 - r)T$$

Where T is the N by N "teleportation" matrix, i.e, the matrix whose entries is 1/N. r is called the damping factor ,with values between zero and one, indicates that random Web surfers move to a different webpage by some means other than selecting a link with probability $1 - r$. Q is still a Markov matrix. However, unlike P, the matrix Q has no zeros. Thus, up to scaling, Q has a unique eigenvector v with eigenvalue 1. We find that eigenvector, normalize it so that the sum of the entries is 1. Then we use v to rank the webpages: the page i with the largest vi is ranked first, and so on.

How to find v (the solution for eigenvalue problem)

Because there are over four billion web pages, so solving $v = Qv$ $(or\ (Q - I)v = 0)$ by Gaussian elimination takes too long. A more efficient method is to divide any nonzero vector whose entries are all nonnegative by the sum of its entries to get a vector w. Then, since Q is a Markov matrix whose columns converge to the steady state eigenvector v, corresponding to the eigenvalue 1, as the power of this matrix approaches infinity, $Q^k w$ converges to v as $k \to \infty$. Therefore, we repeatedly multiply by Q to get Qw, $Q^2 w$, $Q^3 w$, and so on. We keep going until the sequence settles down, i.e. until $Q^{k+1} w$ is nearly equal to $Q^k w$. Then $Q^k w$ is (for all practical purposes) our eigenvector v. In fact, $Qw = rPw + (1 - r)Tw$.
$Tw = u = N$, so

$$Qw = rPw + \frac{1-r}{N} u$$

Additionally, the multiplication Pw can be done rather quickly because the vast majority of the entries of P are 0.

Applying the previous algorithm to 5×5 matrix:

Let

$$Q = \begin{bmatrix} 0.3342 & 0.1580 & 0.1406 & 0.2260 & 0.0422 \\ 0.1113 & 0.2280 & 0.2843 & 0.2099 & 0.3329 \\ 0.0563 & 0.2653 & 0.2813 & 0.3377 & 0.2286 \\ 0.2861 & 0.1206 & 0.0866 & 0.1028 & 0.1698 \\ 0.2121 & 0.2280 & 0.2072 & 0.1235 & 0.2264 \end{bmatrix}$$

Be 5×5 a Markov matrix. To find the eigenvector associated with eigenvalue 1, we should choose a random vector W of size 5×1 whose entries are all nonnegative and the summation of these entries must be 1. Let w be,

$$w= \begin{Bmatrix} 0.1828 \\ 0.2173 \\ 0.1825 \\ 0.2420 \\ 0.1754 \end{Bmatrix}$$

After that, by repeatedly multiplying Qw by Q we get Q^2w, Q^3w, and so on. We stop at a point where $Q^{k+1}w$ is nearly equal to Q^kw.

$$Qw = \begin{Bmatrix} 0.1832 \\ 0.2310 \\ 0.2411 \\ 0.1490 \\ 0.1957 \end{Bmatrix}$$

$$Q^2w = \begin{Bmatrix} 0.1736 \\ 0.2380 \\ 0.2345 \\ 0.1497 \\ 0.2042 \end{Bmatrix}$$

$$Q^3w = \begin{Bmatrix} 0.1710 \\ 0.2397 \\ 0.2361 \\ 0.1488 \\ 0.2044 \end{Bmatrix}$$

$$Q^4w = \begin{Bmatrix} 0.1705 \\ 0.2401 \\ 0.2366 \\ 0.1483 \\ 0.2045 \end{Bmatrix}$$

$$Q^5w = \begin{Bmatrix} 0.1703 \\ 0.2402 \\ 0.2367 \\ 0.1482 \\ 0.2046 \end{Bmatrix}$$

It is clearly shown that the difference between the previous vectors is getting smaller and Q^4w is nearly equal to Q^5w (the summation of the absolute differences between

the elements of these two vectors is 0.0020). So, we can stop at this step by concluding that $Q^5 w = v$ which is the steady state eigenvector.

Applying the previous algorithm to 10×10 matrix:

This part is done by a generic Matlab program that implements the same previous steps of the algorithm to find the steady state eigenvector. This program is attached to this document with the results for an arbitrary 10×10.