**Google Page Rank and Markov Chains**

**W**henever you give a query on Google, you will get the web pages in an order based on their Page Rank. PageRank algorithm is the one, which is behind this ordering of search results.

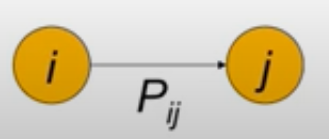
Google page rank is the objective way of rating web pages. We will compute a vector called PageRank Vector defined as the Eigen Vector of the Google Class matrix.

*PageRank Vector: Is the vector that contains PageRank Values.*

**Markov Chain**:

Page rank is based on the random surfer. Let's say we are surfing on the internet by clicking on random links. This can be interpreted as a Markov Chain.

Markov Chain helps in predicting the behavior of the system which is in transition from one state to another by considering only the current state. At each time ‘t’ the system moves from state ‘i’ to ‘j’ with probability **P**ij. **P**ij is called as the transition probability.



The transition probability helps to find out what is the next state of the object by considering only the current state and not any previous ones.

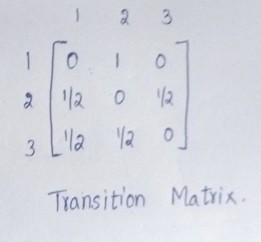
A Markov chain contains:

i) ‘n’ number of states

ii) ‘n x n’ matrix formed from *transition probability.*

Every matrix entry ***P***ij in transition probability matrix(**T**) tells us, P(j|i) the probability of ‘j’ being the next state from the current state ‘i’.

**Transition Matrix Properties**:



i) Each entry in the matrix must be between 0 and 1.

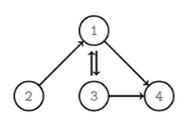
ii) The matrix has to be a square and non-negative matrix.

iii) The Sum of entries in a row has to be equal to 1. This is called as *stochastic matrix*.

**Google Page Rank**:

It’s a popular link-based ranking algorithm. Rather than going into the content and ranking the pages, Page rank makes use of the linked structure to rank the pages.

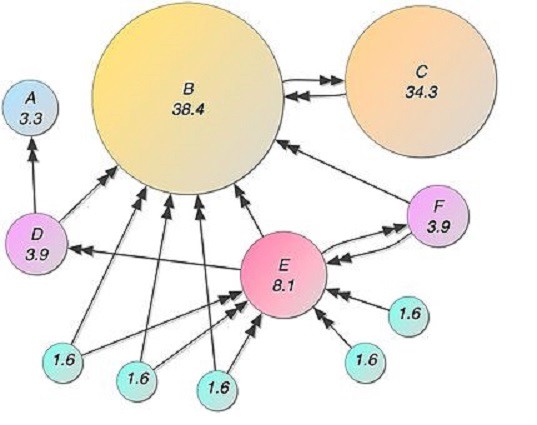
World Wide Web(WWW) is represented as a directed graph W in which all the nodes are pages and edges are hyperlinks. This directed graph is called a Web graph.



A small network with 4 web pages

*Page rank is computed based on the incoming and outgoing links to the page. Link from a high reputed page has a higher weight compared to that of the link from a low reputed page.*

**Example**:



More the incoming links a page has, more important the page is. Even though ‘C’ has a single link from B. The incoming link from B carries more weight. (Image Credits: Wikimedia)

The formula given in the original paper for calculating the page rank is as follows:



PR(A)= PageRank of page A.

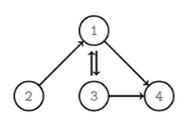
PR(B),PR(C),…. = PageRank of pages [B,C,D….] which link to page A.

L(B),L(C),L(D)…. = Number of outbound links of page .

d = Teleporting (or) damping factor that lies between 0 and 1.

Page rank computation is continued until the page rank gets converged.

Sometimes random surfing leads to dead ends i.e., a page with no outgoing links. These kinds of pages are called **dangling pages** and these pages create storage, computational issues.



Page 4 is the dangling page as it doesn't have any outgoing links

**How to identify the dangling pages in the transition matrix?**

The row corresponding to the dangling page contains all zeros.

**Handling the Dangling Pages**:

**Method\_1**: Connect every dangling page to the hypothetical node of the web graph, construct a self-loop on the hypothetical node.

Constructing the hypothetical node makes a matrix to stochastic, which is necessary for computing the page rank vector because Markov chain is defined only for the stochastic matrix.

**Method\_2**: Replace the row corresponding to the dangling page by **P**ij=1/n for all ‘j’ instead of all zeroes.

By implementing any of the above handling methods we can achieve the stochastic property but it won't guarantee that the Markov model will converge and steady-state vector exists.

The Markov is irreducible(which means every node has to be connected to every other node) but in the real web, every page is not connected to every other page. In order to achieve the irreducibility, all the entries in the transition matrix were made non-zeros i.e., 0<**P**ij<1 to make it regular. This ensures the convergence.

**Google Class Matrix**:

Google matrix is extremely large, it is almost billion by billion in size. So computing the eigen vector for this matrix would be a Herculean task! But there are some numerical methods to compute these eigen vectors as fast as possible.

The google matrix ‘G’ is represented as follows:



P is the matrix from the markov chain. α is the teleporting or damping parameter.

(1-α) is the probability that a random surfer may jump to a random page. α is the probability of clicking the forward link on the current page. Google uses α =0.85 for the PageRank algorithm.

Numerical Methods used for Eigen Vector computation is as follows:

1. Power Method
2. Jacobi Method

After continuously performing some iterations we get the stationary vector.

Reference:

https://medium.com/analytics-vidhya/google-page-rank-and-markov-chains-d65717b98f9c