Nathan Taylor

Eigen See What's Relevant: Explaining the Linear Algebra in the Google PageRank Algorithm

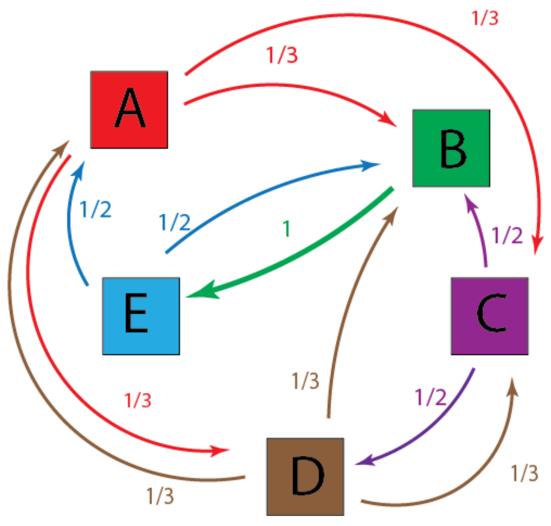
Imagine a typical Google search. You open your internet browser, enter a phrase like “Microsoft” into the search bar and within moments your screen is filled with a list of links to different websites. You don’t have time to scroll through several pages of results to find the website you are looking for, so you want the search engine to provide you with the best information first so you can see it at the top of the list.

Now imagine the same scenario from a website creator’s point of view. You have just made a website to promote your product. Now all you need is for people to visit your website and they are bound to make a purchase. How do you get your potential customers to see and know about your website? You want your site to appear as soon as someone makes a Google search for anything that your webpages talk about so that they will see and visit your website.

This juxtaposition of wills creates an interesting challenge. Users want only the most relevant pages to be displayed at the top of a search result, but every website wants the top spots. If the most visible pages were those that contained the search phrase the most, any website could intentionally add hundreds of common searches hundreds of times to their website and be guaranteed a top spot, regardless of its actual relevance or importance to the user. To combat this, Google developed a system called the PageRank.

Google’s PageRank system assigns a value, called a PageRank, to every page in its network of websites. If a page’s PageRank is high, it will appear earlier in a search result. Each PageRank is calculated by the number of links that point to a given webpage and by the importance of the webpages that point to it.

Consider the following figure, which displays a representation of a network of webpages. Each webpage has links to other webpages, shown here by arrows. Each link carries a fraction of the relevance that the webpage carries. Thus, a webpage with 3 outgoing links will give 1/3 of its importance to each of the webpages it links to. This can also be viewed as the probability of a user clicking on a given link if they are currently in a page containing that link.



This network of links can be represented by a Stochastic or Probability Matrix, where each element is the probability of a link on a given webpage (represented by its column) being selected and taking the user to another webpage (represented by its row). The Stochastic Matrix M for the example network is shown below.

$$\begin{array}{c} A \\ B \\ C \\ D \\ E \end{array} \begin{pmatrix} 0 & 0 & 0 & \frac{1}{3} & \frac{1}{2} \\ \frac{1}{3} & 0 & \frac{1}{2} & \frac{1}{3} & \frac{1}{2} \\ \frac{1}{3} & 0 & 0 & \frac{1}{3} & 0 \\ \frac{1}{3} & 0 & \frac{1}{2} & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Next, to calculate each page's PageRank, we place initial values of 1/n into a vector of size n, where n is the number of pages in the network. In this case, n = 5, so the resulting vector is shown by

$$V1 = \begin{pmatrix} \frac{1}{5} \\ \frac{1}{5} \\ \frac{1}{5} \\ \frac{1}{5} \\ \frac{1}{5} \end{pmatrix}$$

Next, we iteratively multiply matrix M and vector V1 as a Markov Chain. For example, to get the second iteration of V1, called V2, we use M * V1 and get

$$V_2 = \begin{pmatrix} .167 \\ .333 \\ .133 \\ .167 \\ .200 \end{pmatrix}$$

V3 can then be found using M*V2. This process is repeated many times until the values in the vector converge towards specific values and are stable. A faster calculation of V_k is M^k*V1. By selecting larger and larger values k, we see that the values of V_k stabilize at specific values. When k is 1000, the vector V_{1000} has the following values:

$$V_{1000} = \begin{bmatrix} .1875 \\ .3000 \\ .1000 \\ .1125 \\ .3000 \end{bmatrix}$$

Thus, these values correspond to each webpage's PageRank. It is interesting to note that while page B (in green) has 4 different pages pointing to it and page E (in blue) has only 1, these two pages share the same PageRank. This is because E is pointed to by B, which has a large PageRank, so its PageRank gets boosted more than usual. A simplified example of this would be a website that has links pointing to it from 5 different small blogs versus a webpage that is referenced by WebMD. Obviously the more relevant and useful of the two pages is more likely to be the one referenced by WebMD.

An interesting observation to note is that this usage of Markov Chains can be completed through Eigen analysis. Since we are looking for a stable vector so that M*V = V, this can be described as an Eigenvector of M corresponding to the eigenvalue of 1. This eigenvector can be found with the equation $(M-\lambda I)x = 0$. By manipulating the resulting matrix using the toolkit operations until it reaches its Row-Reduced Echelon Form, the general solution x = tv, where v is equal to the PageRank vector shown above.

Therefore, Google periodically "crawls" through its network of webpages to find the number of links pointing to each website. By performing linear algebra and eigen analysis on these values, it can produce a reliable ordering of websites in its list of search results. Because of linear algebra, you and eigen see what is relevant right at the top of our internet searches.