Google PageRank Algorithm

An Application of Linear Algebra

Manish Patel

December 10, 2021

THE INSTITUTE OF SCIENCE, MUMBAI
primespatel@gmail.com

2021-22

Contents

1 Introduction

Humans, the most successful species on the planet, used to pass on their knowledge through books, music, art, and so on. This knowledge transfer continues to this day, but with the most popular, innovative, and easily accessible source that is also a reliable alternative, you guessed it, the Internet. One can conduct a search. What exactly are prime numbers? Google returns n related pages, with the most relevant appearing first.

The process of searching within a document collection for specific information is known as information retrieval (called a query).

An information retrieval challenge for any document collection, especially the Web which concerns precision. The amount of accessible information continues to grow, a user's ability to look at documents does not. Users rarely look beyond the first 10 or 20 documents retrieved. This user impatience means that search engine precision must increase just as rapidly as the number of documents is increasing.

The Web is such a unique document collection which is huge, dynamic, self-organized, and hyperlinked. The Web's self-organization means that, in contrast to traditional document collections, there is no central collection and categorization organization. The web document collection lives in a cyber warehouse, a virtual entity that is not limited by geographical constraints and can grow without limit.

2 Elements of Web Search Engine

2.1 Crawler Module

Web is huge and dynamic in nature as a result, all web search engines have a crawler module. This module contains the software that collects and categorizes the web's documents. The crawling software creates virtual robots, called spiders, that constantly crawls the Web gathering new information and webpages and returning to store them in a central repository.

2.2 Page Repository

The spiders return with new webpages, which are temporarily stored as complete webpages in the page repository. The new pages remain in the repository until they are sent to the indexing module, where their vital information is stripped to create a compressed version of the page.

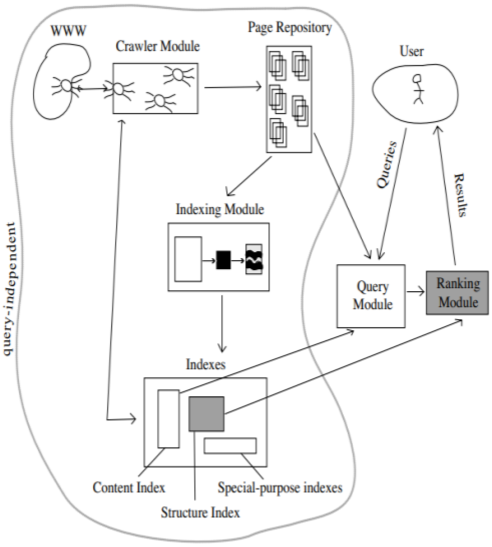


Figure 1: Elements of Web Search Engine

2.3 Indexing Module

The indexing module takes each new uncompressed page and extracts only the vital descriptors, creating a compressed description of the page that is stored in various indexes. The indexing module is like a black box function that takes the uncompressed page as input and outputs a “Cliffnotes” version of the page. The uncompressed page is then tossed out. We will be concentrating in PageRank index in this module.

2.4 Indexes

The indexes hold the valuable compressed information for each webpage. The first is called the content index. Here the content, such as keyword, title, and anchor text for each webpage, is stored in a compressed form using an inverted file structure. The crawler module sometimes accesses the structure index to find uncrawled pages. information regarding the hyperlink structure of pages in the search engine’s index is gleaned during the indexing phase. This link information is stored in compressed form in the structure index. Special-purpose indexes are the final type of index. For example, indexes such as the image index and pdf index hold information that is useful for particular query tasks.

The four modules above (crawler, page repository, indexers, indexes) and their corresponding data files exist and operate independent of users and their queries. Spiders are constantly crawling the Web, bringing back new and updated pages to be indexed and stored. These modules are circled and labeled as query-independent. Unlike the preceding modules, the query module is query-dependent and is initiated when a user enters a query, to which the search engine must respond in real-time.

2.5 Query Module

The query module converts a user’s natural language query into a language that the search system can understand (usually numbers), and consults the various indexes in order to answer the query. For example, the query module consults the content index and its inverted file to find which pages use the query terms. These pages are called the relevant pages. Then the query module passes the set of relevant pages to the ranking module.

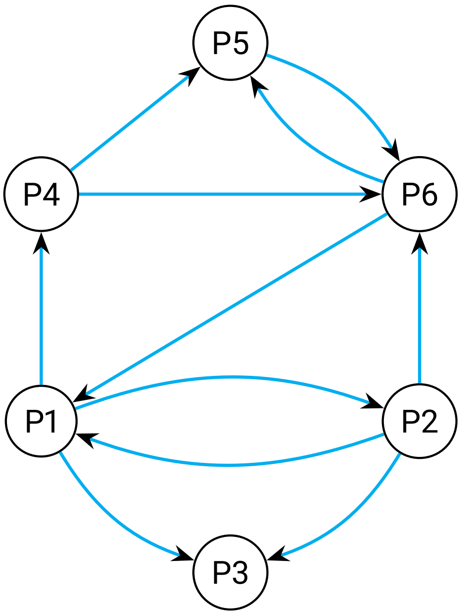
2.6 Ranking Module

The ranking module takes the set of relevant pages and ranks them according to some criterion. The outcome is an ordered list of webpages such that the pages near the top of the list are most likely to be what the user desires. The ranking module is perhaps the most important component of the search process because the output of the query module often results in too many relevant pages that the user must sort through. The set of relevant pages resulting from the query module is then presented to the user in order of their overall scores.

3 PageRank Algorithm

We are interested in calculating PageRank before calculating it, we need to know about the topology of the web.

In order to do that, we need to view the Web as a graph. The Web's hyperlink structure forms a massive directed graph. The nodes in the graph represent webpages and the directed arcs or links represent the hyperlinks. Thus, hyperlinks into a page, which are called inlinks, point into nodes, while outlinks point out from nodes. Nodes with no outlinks are known as dangling node. We will see how it causes the PageRank.



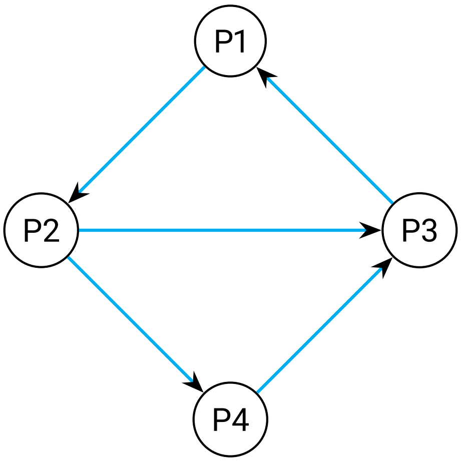
A page is important if it is pointed to by other important pages. Let's define PageRank of a page P_i, denoted $r(P_i)$ and it is the sum of the PageRanks of all pages pointing into P_i.

$$r(P_i) = \sum_{P_j \in B_{P_i}} r(P_j)$$

B_{P_i} : is the set of pages pointing into P_i

The problem with formula is that the $r(P_j)$ values, the PageRanks of pages inlinking to page P_i, are unknown. To sidestep this problem, we are going to use an iterative approach(process). That is, we are assuming that, in the beginning, all pages have equal PageRank (say, $1/n$, here n is the number of web pages). As we are applying the rule in formula successively, we need to introduce some more notation in order to distinguish steps. Let $r_k(P_i)$ be the PageRank of page P_i at k^{th}iteration. Then, PageRank of P_i at $(k+1)^th$ iteration is given by

$$r_{k+1}(P_i) = \sum_{P_j \in B_{P_i}} r_k(P_j)$$



This process is initiated with $r_0(P_i) = 1/n$ for all pages P_i and repeated with the hope that the PageRank scores will eventually converge to some final stable values. Applying equation to the above tiny web of 4-nodes gives the following values for the PageRanks after a few iterations.

Iteration 0	**Iteration 1**	**Iteration 2**	**Iteration 3**
$r_0(P_1) = 1/4$	$r_1(P_1) = 1/4$	$r_2(P_1) = 2/4$	$r_3(P_1) = 2/4$
$r_0(P_2) = 1/4$	$r_1(P_2) = 1/4$	$r_2(P_2) = 1/4$	$r_3(P_2) = 2/4$
$r_0(P_3) = 1/4$	$r_1(P_3) = 2/4$	$r_2(P_3) = 2/4$	$r_3(P_3) = 3/4$
$r_0(P_4) = 1/4$	$r_1(P_4) = 1/4$	$r_2(P_4) = 1/4$	$r_3(P_4) = 1/4$

From the above table we can say that Page P_3 has highest rank and Page P_4 has lowest rank.

This algorithm can be exploited by SEO's (Search Engine Optimization) by just creating dummy websites and pointing(more formally by giving backlinks) to their clients.

This problem can be overcome by little change in previous algorithm.

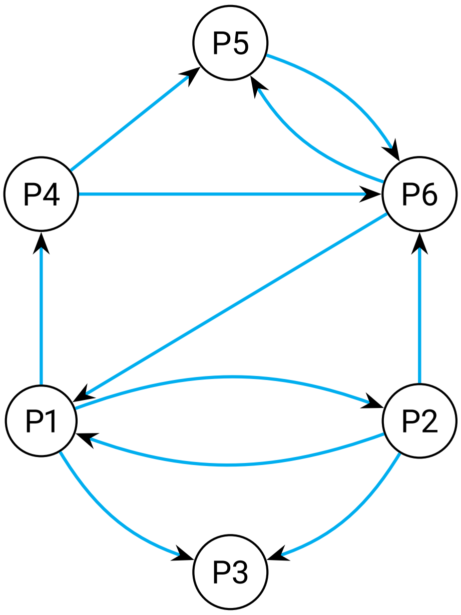
$$r(P_i) = \sum_{P_j \in B_{P_i}} \frac{r(P_j)}{|P_j|}$$

B_{P_i} : is the set of pages pointing into P_i
$|P_j|$: is the number of outlinks from page P_j.

Recursive formula:

$$r_{k+1}(P_i) = \sum_{P_j \in B_{P_i}} \frac{r_k(P_j)}{|P_j|}.$$

Now, unlike previous algorithm dummy websites have no use as their value of backlinks decreases as the number of backlinks increase.

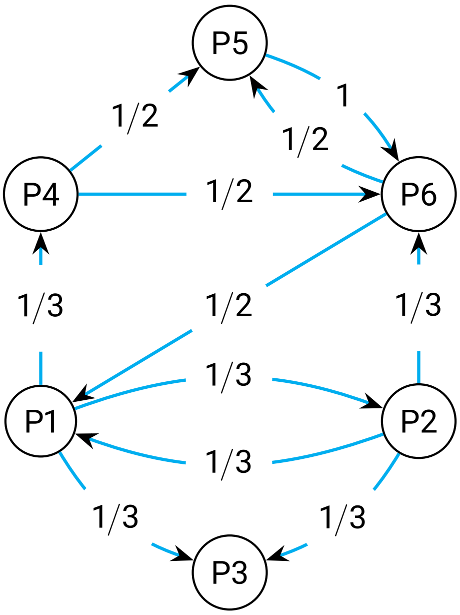


Let's calculate PageRank using updated algorithm,

Iteration 0	Iteration 1	Iteration 2	PageRank
$r_0(P_1) = 1/6$	$r_1(P_1) = 5/36$	$r_2(P_1) = 37/216$	3
$r_0(P_2) = 1/6$	$r_1(P_2) = 2/36$	$r_2(P_2) = 10/216$	5
$r_0(P_3) = 1/6$	$r_1(P_3) = 4/36$	$r_2(P_3) = 14/216$	4
$r_0(P_4) = 1/6$	$r_1(P_4) = 2/36$	$r_2(P_4) = 10/216$	5
$r_0(P_5) = 1/6$	$r_1(P_5) = 6/36$	$r_2(P_5) = 39/216$	2
$r_0(P_6) = 1/6$	$r_1(P_6) = 11/36$	$r_2(P_6) = 46/216$	1

4 Matrix Representation

Equation compute PageRank one page at a time. Using matrices, we replace the tedious $\sum$ symbol, and at each iteration, compute a PageRank vector, which uses a single $1 \times n$ vector to hold the PageRank values for all pages in the index. In order to do this, we introduce an $n \times n$ matrix **H** and a $1 \times n$ row vector $\pi^T \geq 0$ (probability vector i.e. all entries are non-negative real number with sum equal to 1). The matrix **H** is a row normalized hyperlink matrix with $\mathbf{H}_{ij}$ = $1/\,|P_i|$ if there is a link from node i to node j, and 0 otherwise.



Transition Matrix:

$$\begin{array}{ccccccc}
 & P_1 & P_2 & P_3 & P_4 & P_5 & P_6 \\
P_1 & 0 & 1/3 & 1/3 & 1/3 & 0 & 0 \\
P_2 & 1/3 & 0 & 1/3 & 0 & 0 & 1/3 \\
P_3 & 0 & 0 & 0 & 0 & 0 & 0 \\
P_4 & 0 & 0 & 0 & 0 & 1/2 & 1/2 \\
P_5 & 0 & 0 & 0 & 0 & 0 & 1 \\
P_6 & 1/2 & 0 & 0 & 0 & 1/2 & 0
\end{array}$$

Consider last 6-node tiny web again. The **H** matrix for this graph is

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

$$\pi^{(k+1)T} = \pi^{(k)T}\mathbf{H}$$

H is a very sparse matrix (a large proportion of its elements are 0) because most webpages link to few of other pages. $\pi^{(0)T}$ will be row probability vector with all entries $1/n$.

Theorem 4.1: Stochastic matrix

A $n \times n$ Matrix A is said to be row(right) stochastic matrix if

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,j} & \dots & a_{1,n} \\ a_{2,1} & a_{2,2} & \dots & a_{2,j} & \dots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{i,1} & a_{i,2} & \dots & a_{i,j} & \dots & a_{i,n} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \dots & a_{n,j} & \dots & a_{n,n} \end{bmatrix}$$

$$\sum_{j=1}^{n} a_{i,j} = 1$$

$$i = 1, 2, \dots, n$$

H looks a lot like a stochastic transition probability matrix for a Markov chain. The dangling nodes of the network, those nodes with no outlinks, create 0 rows in the matrix. All the other rows, which correspond to the nondangling nodes, create stochastic rows. Thus, We can call H as substochastic.

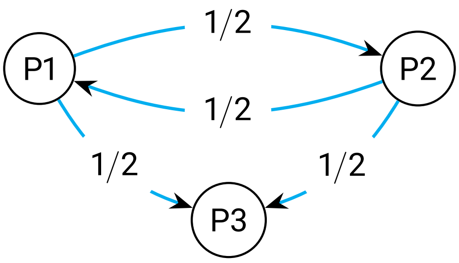
Problem with iterative process iterative equation probably caused readers, especially our mathematical readers, to ask several questions. For example,

- Will this iterative process continue indefinitely or will it converge?
- Under what circumstances or properties of H is it guaranteed to converge?

- Will it converge to something that makes sense in the context of the PageRank problem?
- Will it converge to just one vector or multiple vectors?
- Does the convergence depend on the starting vector $\pi^{(0)T}$?

4.1 Reducibility

There is the problem of rank sinks, those pages that accumulate more and more PageRank at each iteration, monopolizing the scores and refusing to share. Consider the following 3-node graph,



Iteration 0	Iteration 1	Iteration 2	Iteration 3
$r_0(P_1) = 1/3$	$r_1(P_1) = 1/6$	$r_2(P_1) = 1/12$	$r_2(P_1) = 1/24$
$r_0(P_2) = 1/3$	$r_1(P_2) = 1/6$	$r_2(P_2) = 1/12$	$r_2(P_2) = 1/24$
$r_0(P_3) = 1/3$	$r_1(P_3) = 1/3$	$r_2(P_3) = 1/6$	$r_2(P_3) = 1/12$

Here, $\lim\limits_{k\to\infty} \pi^{(0)T}\mathbf{H}^k = 0^T$ In the simple example of 3-node graph, the dangling node P_3 is causing rank sink forcing all PageRank approaching to 0. Thus, ranking pages by their PageRank values is tough when a majority of the nodes are tied with PageRank 0. Ideally, we prefer the PageRank vector should converge to positive probability vector.

Theorem 4.2: Reducible Matrix

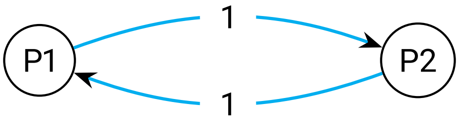
A matrix is reducible if and only if it can be placed into block upper-triangular form by simultaneous row/column permutations.

A square matrix that is not reducible is said to be irreducible.

Rank Sink is caused by the reduciblity of transition matrix H.

4.2 Periodicity

Consider the simplest case of 2-node. Page 1 only points to page 2 and vice versa, creating an infinite loop or cycle.



Suppose the iterative process of equation is run with $\pi^{(0)T} = \begin{pmatrix} 0.25 & 0.75 \end{pmatrix}$. The iterates will not converge no matter how long the process is run. The iterates $\pi^{(k)T}$ flip-flop indefinitely between $\begin{pmatrix} 0.25 & 0.75 \end{pmatrix}$ when k is even and $\begin{pmatrix} 0.75 & 0.25 \end{pmatrix}$ when k is odd.

Theorem 4.3: Eigen Values And Eigen Vectors

For a matrix $\mathbf{A} \in \mathbb{C}^{n \times n}$, the scalars λ and the vectors $\mathbf{x}_{n \times 1} \neq \mathbf{0}$ satisfying $\mathbf{Ax} = \lambda\mathbf{x}$ are the respective eigenvalues and eigenvectors for $\mathbf{A}$. A row vector $\mathbf{y}^T$ is a left-hand eigenvector if $\mathbf{y}^T\mathbf{A} = \lambda\mathbf{y}^T$.

For the above 2-node simple web, it's matrix has eigenvalues 1 & -1 which are of same magnitude.

Theorem 4.4: Spectrum And Spectrum Radius

The set $\sigma(\mathbf{A})$ of distinct eigenvalues is called the spectrum of $\mathbf{A}$, and the spectral radius of $\mathbf{A}$ is the nonnegative number

$$\rho(\mathbf{A}) = \max_{\lambda \in \sigma(\mathbf{A})} |\lambda|.$$

The circle in the complex plane that is centered at the origin and has radius $\rho(\mathbf{A})$ is called the spectral circle

Theorem 4.5: Periodic Matrix

A square matrix A such that the matrix power $A^{k+1} = A$ for some positive integer k, is called a periodic matrix. If k is least such integer, then the matrix is said to have period k. For k = 1 A is idempotent.

If matrix is not periodic then it is aperiodic(k=0). Note: k+1 eigen-values lie on the spectral circle.
Periodicity of the transition matrix H is resulting in problem of cycle.

Therefore, the PageRank convergence problems caused by sinks and cycles can be overcome if H is modified slightly so that it is a Markov matrix with these desired properties to get desired output. Further, If our Google matrix some how become aperiodic then we can get PageRank vector just by applying power method with iteration formula regardless of the starting probability vector.
Now, we know that for any starting probability vector, the power method applied to a Markov matrix M converges to a unique positive vector called the stationary vector as long as P is stochastic, irreducible, and aperiodic.

4.3 Stochasticity Adjustment

Imagine a web surfer who bounces along randomly following the hyperlink structure of the Web. In the long run, the proportion of time the random surfer spends on a given page is a measure of the relative importance of that page. That is, when he arrives at a page with several outlinks, he chooses one at random, hyperlinks to this new page, and continues this random decision process indefinitely. This random surfer encounters some problems. He gets caught whenever he enters a dangling node. And on the Web there are plenty of nodes dangling, e.g., pdf files, image files, data tables, etc.
The 0^T rows of H are replaced with 1/n e^T , thereby making H stochastic. As a result, the random surfer, after entering a dangling node, can now hyperlink to any page at random.

For the tiny 6-node web, Our transition matrix H can be made stochastic matrix called S by simply making RankOne update to H

$$\mathbf{S} = \begin{bmatrix} 0 & 1/3 & 1/3 & 1/3 & 0 & 0 \\ 1/3 & 0 & 1/3 & 0 & 0 & 1/3 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 0 & 0 & 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1/2 & 0 & 0 & 0 & 1/2 & 0 \end{bmatrix}$$

For any transition matrix H, stochastic matrix S can be calculated using following formula

$$\mathbf{S} = \mathbf{H} + \mathbf{a}\left(1/n\mathbf{e}^T\right)$$

where $\mathbf{e}^T$ is row vector with all entries 1
a is binary column vector called dangling node vector.
Entries of **a**, a_i = 1 if page i is a dangling node and 0 otherwise.
This adjustment guarantees that **S** is stochastic, and thus, is the transition probability matrix for a Markov chain. However, it alone cannot guarantee the convergence results we desired. (That is, that a unique positive π^T exists)

4.4 Primitivity Adjustment

A primitive matrix is both irreducible and aperiodic. Thus, the PageRank vector of the chain exists, is unique, and can be found by a simple power iteration.

Theorem 4.6: Positive Matrix

A matrix P is positive if every entry is positive, we write $P > 0$ and matrix P is nonnegative if every entry is zero or positive, we write $P \geq 0$.
Primitive Matrix: A square non-negative real matrix P is said to be primitive if there exists a positive integer k such that $P^k > 0$.

The random surfer argument for the primitivity adjustment goes like this. While it is true that surfers follow the hyperlink structure of the Web, at times they get bored and abandon the hyperlink method of surfing by entering a new destination in the browser's URL line. When this happens, the random surfer, like a Star Trek character, "teleports" to the new page, where he begins hyperlink surfing again, until the next teleportation, and so on.

To model this mathematically, consider new matrix **G**, such that

$$\mathbf{G} = \alpha\mathbf{S} + (1-\alpha)1/n\mathbf{e}\mathbf{e}^T,$$

where α is a scalar between 0 and 1. G is called the Google matrix. In this model, α is a parameter that controls the proportion of time the random surfer follows the hyperlinks as opposed to teleporting. Suppose α = 0.8. Then 80% of the time the random surfer follows the hyperlink structure of the Web and the other 20% of the time he teleports to a random new page. The teleporting is random because the teleportation matrix **E** = $1/n$ ee ^{T} is uniform, meaning the surfer is equally likely, when teleporting, to jump to any page.

There are several consequences of the primitivity adjustment:

Matrix G is the convex combination of the two stochastic matrices **S** and **E** = $1/n\mathbf{e}\mathbf{e}^T$ hence it is stochastic.

Theorem 4.7: Norm

In the applications involving PageRank and Markov chains, it's more natural (and convenient) to use the vector 1 -norm defined by

$$\|\mathbf{x}\|_1 = \sum_{i=1}^{n} |x_i|$$

Theorem 4.8: Perron-Frobenius Theorem

If $\mathbf{A}_{n\times n} \geq \mathbf{0}$ is irreducible, then each of the following is true.
1. $r = \rho(\mathbf{A}) > 0$.
2. $r \in \sigma(\mathbf{A})$ (r is the Perron root).
3. alg mult $_{\mathbf{A}}(r) = 1$. (the Perron root is simple).
4. There exists an eigenvector $\mathbf{x} > \mathbf{0}$ such that $\mathbf{Ax} = r\mathbf{x}$.
5. The Perron vector is the unique vector defined by

$$\mathbf{Ap} = r\mathbf{p}, \quad \mathbf{p} > \mathbf{0}, \|\mathbf{p}\|_1 = 1,$$

and, except for positive multiples of **p**, there are no other nonnegative eigenvectors for **A**, regardless of the eigenvalue.
6. r need not be the only eigenvalue on the spectral circle of **A**.

G is primitive because $\text{G}^k > 0$ for some k. (In fact, this holds for $k = 1$.) G is irreducible as well as aperiodic. G is primitive so irreducibility and aperiodicity are trivially enforced. By Perron-Frobenius Theorem there exists a unique positive probability vector π^T exists, and it can be obtained by applying power method to G.
G is completely dense, which is a very bad thing, computationally. Fortunately, G can be written as a rank-one update to the very sparse hyperlink matrix **H**. This is computationally advantageous.

$$\begin{aligned}\mathbf{G} &= \alpha\mathbf{S} + (1-\alpha)1/n\mathbf{e}\mathbf{e}^T \\ &= \alpha\left(\mathbf{H} + 1/n\mathbf{a}\mathbf{e}^T\right) + (1-\alpha)1/n\mathbf{e}\mathbf{e}^T \\ &= \alpha\mathbf{H} + (\alpha\mathbf{a} + (1-\alpha)\mathbf{e})1/n\mathbf{e}^T.\end{aligned}$$

In summary, Google's adjusted PageRank method is

$$\pi^{(k+1)T} = \pi^{(k)T}\mathbf{G},$$

which is simply the power method applied to G.