Page Rank

Katarina Fabek, Goran Lalić, Ivan Leverić

December 18, 2020

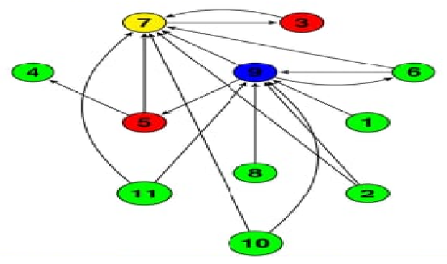
Contents

1 Abstract

We present an algorithm for calculating the PageRank of the Google matrix G. The algorithm lumps all dangling nodes into one node and we will see that lumping is a similarity transformation of G. We will study the PageRank of nondangling nodes separately from that of dangling nodes. We also apply the power method to the smaller lumped matrix and observe convergence rate. Also, if we increase the number of dangling nodes, the efficiency of the algorithm also increases. We will observe the case when we want to distinguish different classes of dangling nodes.
We show that the PageRank of the dangling nodes depends on that of the nondangling nodes. Also, we will observe the case when all web pages are dangling nodes and the storage issue that concerns dangling nodes.

2 Introduction: How did Google beat other search engines?

In the late 1990s, when Google was released, one thing made it special from other search engines: those better, higher-quality pages were listed first. But how did Google do that? How did he arrange the pages, noticed which ones were of better quality than the others? Here comes Page Rank, the algorithm used by Google to rank pages. Therefore, the idea is as follows. We attach importance (weight) to each page, a number that is greater than or equal to zero. But what is the importance of the site in general? The web site is as important as the ones that point to it. More precisely, let us represent everything by a directed graph. Let the nodes of the graph represent the web pages and the directed edges represent a link from one page to another.



Picture 2.1.

The importance of nodes decreases in proportion to the number of outbound links from that page. It turns out that the importance or weight of the nodes is a component of the eigenvector we want to determine. Therefore, page ranking depends on the components of the eigenvector of stochastic google matrix G.

The Google matrix G is a convex combination of two stochastic matrices S and E,

$$G = \alpha S + (1-\alpha)E \text{ for } \alpha \in [0, 1\rangle$$

where S represents structure of the web, so element (i, j) of S is nonzero if there exist a link from i to j, and matrix E is a matrix of rank one which ensures the uniqueness of the PageRank vector.
But, not all web pages contain links to other pages. These pages are called dangling nodes. So, the rows in matrix S which represents dangling nodes are zero rows so we replace these rows with the same vector w so matrix S would be stochastic.
Further, we will study lumping because we want to exclude dangling nodes from PageRank algorithm.

3 Google matrix G and the PageRank

Let n be the number of web pages, and k the number of nondangling nodes, $1 \leq k < n$. First we have $n \times n$ matrix P that represents the link structure of the web. Element p_{ij} of matrix P is the probability of moving from page i to page j in one time-step. Matrix P has $n-k$ zero rows that denote dangling nodes, so first we switch all $n-k$ zero rows to the end to get $n \times n$ matrix H:

$$H = \begin{bmatrix} H_{11} & H_{12} \\ 0 & 0 \end{bmatrix}$$

where $n-k$ zero rows also denote dangling nodes, $k \times k$ matrix H_{11} represents the links that connect nondangling nodes and H_{12} represents the links that connect nondangling to dangling nodes. We have

$$H_{11} \geq 0 \ , \ H_{12} \geq 0 \quad \text{elementwise}$$

$$H_{11}e + H_{12}e = e \tag{1}$$

where e is the column vector with all components equal to one, $e = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}$.

We want that H is a stochastic matrix, but the problem is $n-k$ zero rows of the matrix.

One way we can solve this is by using a vector which maximal column sum is equal to one. So, idea is to replace each zero row by the same *dangling node vector* w

$$w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}, \quad w \geq 0,$$

$$\|w\| = w^T e = 1 \tag{2}$$

where w_1 is $k \times 1$, w_2 is $(n-k) \times 1$.
Let $d = \begin{bmatrix} 0 \\ e \end{bmatrix}$.
Then the matrix

$$\begin{aligned} S = H + dw^T &= \begin{bmatrix} H_{11} & H_{12} \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 \\ e \end{bmatrix} \begin{bmatrix} w_1^T & w_2^T \end{bmatrix} \\ &= \begin{bmatrix} H_{11} & H_{12} \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ ew_1^T & ew_2^T \end{bmatrix} \\ &= \begin{bmatrix} H_{11} & H_{12} \\ ew_1^T & ew_2^T \end{bmatrix} \end{aligned}$$

$S \geq 0$ and, according to (1) and (2), $Se = e$. So, S is stochastic.

Let's remember, matrix G is $G = \alpha S + (1-\alpha)E$, where both S and E are stochastic matrices.
Instead of the previously mentioned matrix E, we can use $E = ev^T$, where v is a probability vector called the *personalization or teleportation vector*, such that

$$v = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}, \quad v \geq 0,$$

$$||v|| = 1$$

where v_1 is $k \times 1$, v_2 is $(n-k) \times 1$.

Then stochastic Google matrix G is defined as the convex combination

$$G = \alpha S + (1-\alpha) ev^T, \quad \text{for } \alpha \in [0, 1\rangle \tag{3}$$

Using infinity norm of matrix G, for an arbitrary eigenvalue λ_i and its belonging eigenvector z

$$|\lambda_i| \, ||z|| = ||\lambda_i z|| = ||Gz|| \leq ||G|| \, ||z||$$

So, $|\lambda_i| \leq ||G|| = 1$ which means that eigenvalues $\lambda_1 = \lambda = 1$ is dominant eigenvalue

$$|\lambda_i| \leq |\lambda_1| \text{ for every } i = 2, ..., n$$

According to (3), for eigenvalues $\lambda_1 = 1, \lambda_2(S), ..., \lambda_n(S)$ of matrix S, eigenvalues of matrix G are

$$1, \alpha\lambda_2(S), ..., \alpha\lambda_n(S)$$

Knowing that eigenvalue $\lambda = 1$ have geometric multiplicity equal to one and the magnitude of all other eigenvalues is bounded by α, G has unique stationary distribution π

$$\pi^T G = \pi^T \ , \quad \pi \geq 0,$$
$$||\pi|| = 1$$

and π is called PageRank. Element k of π represents the PageRank for web page k.
Equally, instead of $E = ev^T$, we can use matrix with all elements equal to $\frac{1}{n}$, $E = \frac{1}{n}ee^T$. This means that the teleportation probabilities are uniformly distributed. So, G is a convex combination of two stochastic matrices which insure that G is both stochastic and irreducible, because every node is connected directly to every other node. This also means that G is primitive, so in this case, G also has unique stationary distribution π.

Now let's partition the PageRank like we did with G

$$\pi = \begin{bmatrix} \pi_1 \\ \pi_2 \end{bmatrix}$$

π_1 represents the PageRank associated with the nondangling nodes and π_2 represents the PageRank of the dangling nodes.

4 Lumping

The Google matrix represents a lumpable Markov chain. Idea of lumpability is to reduce the size of the state space of some continuous-time Markov chains. Let's take some example. Knowing that stochastic matrix P is a lumpable matrix on a partition t if and only if, for any subsets t_i and t_j in the partition, and for any states n, m in subset t_i

$$\sum_{z \in t_j} p(n,z) = \sum_{z \in t_j} p(m,z)$$

where $p(i,k)$ is the probability of moving from state i to state k.

For example,

$$P = \begin{bmatrix} \frac{1}{2} & \frac{1}{4} & \frac{1}{8} & \frac{1}{8} \\ \frac{5}{8} & \frac{1}{8} & \frac{1}{4} & 0 \\ 0 & \frac{1}{8} & \frac{7}{8} & 0 \\ \frac{1}{8} & 0 & \frac{5}{8} & \frac{1}{4} \end{bmatrix}$$

This matrix is lumpable on the partition $t = \{(1,2),(3,4)\}$ so matrix

$$P_t = \begin{bmatrix} \frac{3}{4} & \frac{1}{4} \\ \frac{1}{8} & \frac{7}{8} \end{bmatrix}$$

is lumped matrix of P on t.
Let's go back to the PageRank.
Lumpability in matrix terms with a permutation matrix P and

$$PMP^T = \begin{bmatrix} M_{11} & ... & M_{1,k+1} \\ \vdots & & \vdots \\ M_{k+1,1} & ... & M_{k+1,k+1} \end{bmatrix}$$

a partition of a stochastic matrix M. Then M is lumpable with respect to this partition if each vector $M_{ij}e$ is a multiple of the vector e that has all components equal to one, $i \neq j$, $1 \leq i,j \leq k+1$.

For our example, if we put

$$M_{11} = \begin{bmatrix} \frac{1}{2} & \frac{1}{4} \\ \frac{5}{8} & \frac{1}{8} \end{bmatrix}, \quad M_{12} = \begin{bmatrix} \frac{1}{8} & \frac{1}{8} \\ \frac{1}{4} & 0 \end{bmatrix}$$

$$M_{21} = \begin{bmatrix} 0 & \frac{1}{8} \\ \frac{1}{8} & 0 \end{bmatrix}, \quad M_{22} = \begin{bmatrix} \frac{7}{8} & 0 \\ \frac{5}{8} & \frac{1}{4} \end{bmatrix}$$

So, $M_{11}e = \frac{3}{4}e$, $M_{12}e = \frac{1}{4}e$, $M_{21}e = \frac{1}{8}e$ and $M_{22}e = \frac{7}{8}e$.

For Google matrix G, if all dangling nodes are lumped into a single node, then G is lumpable.
We can write G as

$$G = \begin{bmatrix} G_{11} & G_{12} \\ eu_1^T & eu_2^T \end{bmatrix} \tag{4}$$

where

$$u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \alpha w + (1-\alpha)v$$

and G_{11} is $k \times k$, and G_{12} is $(n-k) \times k$.
So, element (i, j) of G_{11} corresponds to block M_{ij} , $1 \leq i, j \leq k$; row i of G_{12} corresponds to block $M_{i,k+1}$, $1 \leq i \leq k$; column i of eu_1^T corresponds to $M_{k+1,i}$, $1 \leq i \leq k$; and eu_2^T corresponds to $M_{k+1,k+1}$.
Lumping the dangling nodes is a similarity transformation of the Google matrix, so we lump all dangling nodes into a single node and that can be done by a similarity transformation. That transformation reduces Google matrix G to block upper triangular form

$$\begin{bmatrix} G^{(1)} & * \\ 0 & 0 \end{bmatrix} \text{ where } G^{(1)} = \begin{bmatrix} G_{11} & G_{12}e \\ u_1^T & u_2^T e \end{bmatrix}$$

The matrix $G^{(1)}$ is stochastic of order $k+1$ and has the same nonzero eigenvalues as matrix G.

5 Algorithm for PageRank

Before describing an algorithm, let's look stationary distribution σ of the smaller matrix $G^{(1)}$:

Theorem 1.: With the notation in section 3 and the matrix G as partitioned in (4), let

$$\sigma^T \begin{bmatrix} G_{11} & G_{12}e \\ u_1^T & u_2^T e \end{bmatrix} = \sigma^T, \quad \sigma \geq 0,$$
$$||\sigma|| = 1$$

and partition $\sigma^T = \begin{bmatrix} \sigma_{1:k}^T & \sigma_{k+1} \end{bmatrix}$, where σ_{k+1} is scalar. Then the PageRank equals

$$\pi^T = \begin{bmatrix} \sigma_{1:k}^T & \sigma^T \begin{pmatrix} G_{12} \\ u_2^T \end{pmatrix} \end{bmatrix}.$$

Now we can present an algorithm based on **Theorem 1.**. We will compute the PageRank π from the stationary distribution σ of the lumped matrix $G^{(1)}$.

The inputs to the algorithm are: the matrix H, the personalization vector v, the dangling node vector w, and α.

The output of the algorithm is an approximation $\hat{\pi}$ of the PageRank π, which is computed from an approximation $\hat{\sigma}$ of σ.

Algorithm:

```
% Inputs: H, v, w, $\alpha$; Output: $\hat{\pi}$
% We applied a Power method to $G^{(1)}$:
Choose a starting vector $\hat{\sigma}^T = \begin{bmatrix} \hat{\sigma}_{1:k}^T & \hat{\sigma}_{k+1} \end{bmatrix}$, where $\hat{\sigma} \geq 0, ||\hat{\sigma}|| = 1$
While not converged
$\hat{\sigma}_{1:k}^T = \alpha \hat{\sigma}_{1:k}^T H_{11} + (1-\alpha) v_1^T + \alpha \hat{\sigma}_{k+1} w_1^T$
$\hat{\sigma}_{k+1} = 1 - \hat{\sigma}_{1:k}^T e$
end while

% We need to recover PageRank:
$\hat{\pi}^T = \begin{bmatrix} \hat{\sigma}_{1:k}^T & \alpha \hat{\sigma}_{1:k}^T H_{12} + (1-\alpha) v_2^T + \alpha \hat{\sigma}_{k+1} w_2^T \end{bmatrix}$
```

The convergence rate of the power method applied to G is α. This algorithm has the same convergence rate, because $G^{(1)}$ has the same nonzero eigenvalues as G, according to similarity transformation. But, in this case, convergence rate is much faster because we operate with smaller matrix because dangling nodes are excluded from power method. This is achieved because each iteration of the

power method (applied to $G^{(1)}$) includes a sparse matrix vector multiply with $k \times k$ matrix H_{11} and some vector operations. So, the operation count depends on the number of nondangling nodes and not on the total number of web pages. In last step, π is recovered, and in this case a sparse matrix vector is multiplied with the $k \times (n-k)$ matrix H_{12} and we also have some vector operations.
It can be concluded that this algorithm is faster than power method applied to G, but it is still simple and use minimal storage (more about storage in section 8.).

5.1 Several dangling node vectors

In this section, we won't be treating all dangling nodes in the same way. This time, we will assign them different dangling node vectors. For example, if we want to distinguish dangling nodes based on their function (text files, images, videos, songs...) or their topic or something else. In this case, one dangling node vector wouldn't be helpful.
So, we have $m \geq 1$ different classes of dangling nodes and we assign a different dangling node vector w_i to each class, for every $i = 1, ..., m$.
Now we need to replace $n-k$ zero rows in matrix H by $m \geq 1$ different dangling node vetors $w_1, ..., w_m$.
Considering this, the more general Google matrix is

$$\text{F=}\quad \begin{array}{c c} & \begin{array}{cccc} k & k_1 & ... & k_m \end{array} \\ \begin{array}{c} k \\ k_1 \\ \vdots \\ k_m \end{array} & \begin{bmatrix} F_{11} & F_{12} & ... & F_{1,m+1} \\ eu_{11}^T & eu_{12}^T & ... & eu_{1,m+1}^T \\ \vdots & \vdots & & \vdots \\ eu_{m,1}^T & eu_{m,2}^T & ... & eu_{m,m+1}^T \end{bmatrix} \end{array}$$

$$\text{where } u_i = \begin{bmatrix} u_{i,1} \\ \vdots \\ u_{i,m+1} \end{bmatrix} = \alpha w_i + (1-\alpha)v$$

If we put $\tilde{\pi}$ to be PageRank of F,

$$\tilde{\pi}^T F = \tilde{\pi}^T \ , \quad \tilde{\pi} \geq 0,$$
$$||\tilde{\pi}|| = 1.$$

Now we can extend our **Theorem 1.** to any number m of dangling node vectors.

Theorem 2.: Let ρ be the stationary distribution of the lumped matrix

$$F^{(1)} = \begin{bmatrix} F_{11} & F_{12}e & ... & F_{1,m+1}e \\ u_{11}^T & u_{12}^T e & ... & u_{1,m+1}^T e \\ \vdots & \vdots & & \vdots \\ u_{m,1}^T & u_{m,2}^T e & ... & u_{m,m+1}^T e \end{bmatrix},$$

that is,

$$\rho^T F^{(1)} = \rho^T, \quad \rho \geq 0, \quad ||\rho|| = 1.$$

With the partition $\rho^T = \begin{bmatrix} \rho_{1:k}^T & \rho_{k+1:k+m}^T \end{bmatrix}$, where $\rho_{k+1:k+m}$ is $m \times 1$, the PageRank of F equals

$$\tilde{\pi}^T = \begin{bmatrix} \rho_{1:k}^T & \rho^T \begin{pmatrix} F_{12} & ... & F_{1,m+1} \\ u_{12}^T & ... & u_{1,m+1}^T \\ \vdots & & \vdots \\ u_{m,2}^T & ... & u_{m,m+1}^T \end{pmatrix} \end{bmatrix}.$$

6 PageRank of dangling vs. nondangling nodes

In this section, we observe how PageRanks of nondangling nodes π_1 and PageRanks of dangling nodes π_2 influence each other. Also, we examine the effect of the dangling node vector w on π_1 and π_2.
Let's remember, we partition the PageRank π like we did with Google matrix G:

$$\pi = \begin{bmatrix} \pi_1 \\ \pi_2 \end{bmatrix}$$

where π_1 represents the PageRank of the nondangling nodes and π_2 represents the PageRank of the dangling nodes.
Also, dangling node vector w and personalization vector v

$$w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}, \quad v = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

w_1 and v_1 are $k \times 1$ and w_2 and v_2 are $(n-k) \times 1$, where k is the number of nondangling nodes and n is the number of all web pages. We also had $k \times k$ matrix H_{11}, representing the links that connect nondangling nodes, and H_{12}, representing the links that connect nondangling to dangling nodes.

According to **Algorithm**, we can see that π_2 does not affect the calculation of PageRank of the nondangling nodes π_1. So, π_1 can be computed separately from π_2. On the other hand, π_1 affects the calculation of π_2.
We have

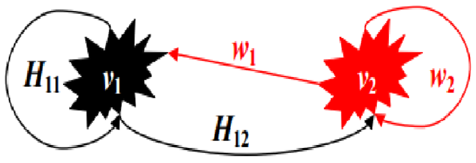
$$\pi_1^T = ((1-\alpha)v_1^T + \rho w_1^T)(I - \alpha H_{11})^{-1} \tag{5}$$

$$\pi_2^T = \alpha \pi_1^T H_{12} + (1-\alpha)v_2^T + \alpha(1 - \|\pi_1\|)w_2^T \tag{6}$$

where

$$\rho = \alpha \frac{1 - (1-\alpha)v_1^T (I - \alpha H_{11})^{-1} e}{1 + \alpha w_1^T (I - \alpha H_{11})^{-1} e} \geq 0$$

and I is the identity matrix and e is all-ones vector.
We conclude that π_2 depends directly on π_1.



Picture 6.2.

So, π_1 does not depend on w_2 (the connectivity among the dangling nodes), v_2 (the personalization vector for the dangling nodes) and H_{12} (the links that connect nondangling to dangling nodes). But, we have the following dependence:

$$\|v_1\| + \|v_2\| = 1$$
$$\|w_1\| + \|w_2\| = 1$$
$$H_{11}e + H_{12}e = e$$

Therefore we have the dependence through norms and not on individual elements of w_2, v_2 and H_{12}. So, π_1 does not depend on π_2. Also, the PageRank of the nondangling nodes does not depend on number of dangling nodes, because it can be computed without any knowledge of the PageRank of the dangling

nodes.
From the formulas given above, it's easy to see that the PageRank of the nondangling nodes π_1 depends on w_1 and v_1, distributed through the links H_{11}. On the other hand, the PageRank of the dangling nodes π_2 comes from w_2, v_2 and π_1 through the links H_{12}. Therefore, H_{12} has a major role in how much the PageRank of the nondangling nodes affects the calculation of π_2. Also, according to (6), the influence of w_2 on π_2 decreases as the combined PageRank $\|\pi_1\|$ of nondangling nodes increases (because then (1-$\|\pi_1\|$) decreases).

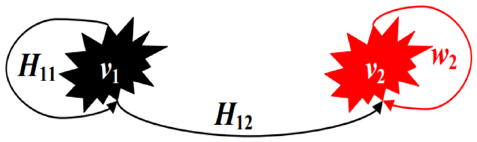
Let's take a look on the effect of the dangling node vector w on π_1 and π_2. We take norm $\|w\| = w^T e$, $w \geq 0$ in (5) and denote $\|w\|_H = w^T(I - \alpha H_{11})^{-1}e$:

$$\begin{aligned}
\pi_1^T e &= ((1-\alpha)v_1^T + \rho w_1^T)(I - \alpha H_{11})^{-1}e \\
\|\pi_1\| &= (1-\alpha)v_1^T(I-\alpha H_{11})^{-1}e + \rho w_1^T(I - \alpha H_{11})^{-1}e \\
\|\pi_1\| &= (1-\alpha)\|v_1\|_H + \rho\|w_1\|_H \\
\|\pi_1\| &= (1-\alpha)\|v_1\|_H + \alpha\frac{1-(1-\alpha)v_1^T(I-\alpha H_{11})^{-1}e}{1+\alpha w_1^T(I-\alpha H_{11})^{-1}e}\|w_1\|_H \\
\|\pi_1\| &= (1-\alpha)\|v_1\|_H + \alpha\frac{1-(1-\alpha)\|v_1\|_H)^{-1}e}{1+\alpha\|w_1\|_H}\|w_1\|_H \\
\|\pi_1\| &= \frac{(1-\alpha)\|v_1\|_H(1+\alpha\|w_1\|_H) + \alpha\|w_1\|_H - \alpha\|w_1\|_H(1-\alpha)\|v_1\|_H}{1+\alpha\|w_1\|_H}
\end{aligned}$$

$$\Rightarrow \boxed{\|\pi_1\| = \frac{(1-\alpha)\|v_1\|_H + \alpha\|w_1\|_H}{1+\alpha\|w_1\|_H}}$$

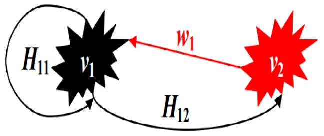
Therefore, the combined PageRank $\|\pi_1\|$ of nondangling nodes is an increasing function of $\|w_1\|$, because $(1-\alpha)\|w_1\| \leq \|w_1\|_H \leq \frac{1}{1-\alpha}\|w_1\|$.

Since $\|w_1\| \geq 0$, the combined PageRank $\|\pi_1\|$ is minimal when $w_1 = 0$. Then the nondangling nodes receive their PageRank π_1 only from v_1, but the dangling nodes receive it from v_2, w_2 and from the PageRank of the nondangling nodes through the links H_{12}. In that case, the dangling node vector w_2 has a stronger influence on π_2.



Picture 6.2. Page Rank when $w_1 = 0$.

On Picture 6.2., we can see that there is no links back from dangling to non-dangling nodes, so the dangling nodes absorb more PageRank.
When $w_2 = 0$, then the dangling nodes receive their PageRank π_2 only from v_2 and again from the PageRank of the nondangling nodes through the links H_{12}.



Picture 6.3. Page Rank when $w_2 = 0$.

The nondangling nodes receive their PageRank π_1 from v_1 and from w_1 distributed through the links H_{11}. On Picture 6.3., we can see that there is no links between dangling nodes. From (6) we obtain

$$\pi_2^T = \alpha\pi_1^T H_{12} + (1-\alpha)v_2^T$$

So, π_1 affects the calculation of π_2, but in this case, it only has a positive influence because there is no part with 1-$\|\pi_1\|$.

Last case, when a dangling node vector w and personalization vector v are the same, the PageRank vector π is a multiple of $v^T(I-\alpha H)^{-1}$:

$$\pi^T = \frac{1-\alpha}{1-\alpha v^T(I-\alpha H)^{-1}d} v^T(I-\alpha H)^{-1}$$

where $d = \begin{bmatrix} 0 \\ e \end{bmatrix}$.

7 Only dangling nodes

In this section, we will observe the case when all web pages are dangling nodes. Let's go back to Google matrix G. Again, let n be the number of web pages, which are all dangling nodes. In section 2, we noted that the Google matrix G is a convex combination

$$G = \alpha S + (1-\alpha)E \text{ for } \alpha \in [0,1\rangle$$

where S represent structure of the web, so element (i,j) of S is nonzero if there exist a link from i to j. We constructed matrix S from matrix H, but in this case, matrix H is a zero matrix because n zero rows denote n dangling nodes, so we need to replace each zero row by the same dangling node vector w, where $w \geq 0$, $\|w\| = w^T e = 1$. So, $S = ew^T$, S is stochastic and has rank one. According to that, the Google matrix G has also rank equal to one and taking a personalization vector v, $v \geq 0$, $\|v\| = v^T e = 1$, we obtain

$$G = \alpha e w^T + (1-\alpha) e v^T, \quad \text{for } \alpha \in [0,1\rangle$$

We can also write the Google matrix as

$$G = eu^T$$

where

$$u = \alpha w + (1-\alpha) v.$$

Let vector e_j denotes the jth column of matrix I and let $u_j \neq 0$ be a nonzero element of u. Then $1 = u^T e \neq 0$, so $G = eu^T$ is diagonalizable:

$$X^{-1} G X = e_j e_j^T$$

where $X^{-1} = I - e_j e_j^T - eu^T + 2 e_j u^T$.

Multiplying on the right by matrix X^{-1} and then on the left by e_j^T and using $XX^{-1} = I$ and $e_j^T e_j = I$, we obtain

$$e_j^T X^{-1} G = e_j^T X^{-1}$$

Knowing that $\pi^T G = \pi^T, \quad \pi \geq 0, \quad \|\pi\| = 1$, we get

$$\pi^T = e_j^T X^{-1} = u^T.$$

8 Storage issues

The problem we may encounter with this algorithm is the storage of the matrix P in main memory. For small subsets of the web, when matrix P fits in main memory, computation of the PageRank can be implemented as usual. But, when P does not fit, we have two options: compress the data to fit in main memory and implement a modified version of PageRank on that data, or keep the data in its uncompressed form and develop I/O-efficient implementations that must take place on the large, uncompressed data.
In the case when P fits in main memory, compressing the data is not necessary, but still some techniques should be applied to reduce the work involved in each iteration.
We can decompose matrix P

$$P = D^{-1} A \tag{7}$$

where D is diagonal matrix holding outdegrees of the nodes and the adjacency matrix A of 0s and 1s. This decomposition is very useful in saving storage and reducing the number of multiplications in each vector-matrix multiplication $x^T P$ in the power method.

If $nnz(P)$ is the number of elements in P that are not zero and we don't have decomposition given in (7), the power method requires $nnz(P)$ multiplications and $nnz(P)$ additions. On the other hand, with this decomposition

$$x^T P = x^T D^{-1} A = (x^T) \cdot (diag(D^{-1}))A$$

where we apply component-wise multiplication of the elements in vectors x^T and $diag(D^{-1})$, which requires n multiplications. Since A is an adjacency matrix, $[(x^T) \cdot (diag(D^{-1}))]A$ requires and $nnz(P)$ additions as well. So, it saves us $nnz(P) - n$ multiplications.
Idea of web-sized implementations of the PageRank is to store P or A in an adjacency list of the columns of the matrix to speed up the PageRank power method, because at each iteration k, it requires vector-matrix multiplications of $x^{(k-1)T} P$.

So, column i contains information about which nodes have a link on a page i, which is important information for ranking pages with the PageRank algorithm.

The storage issue we observe concerns dangling nodes.
Let's remember, to create the stochastic Google matrix G, for matrix E we used matrix with all elements equal to $\frac{1}{n}$ or $E = ev^T$. In this case, storage is a problem, so we can fix this with vector a and also have the stochastic matrix.

Element $a_i = 1$ if row i of P represent a dangling node. Otherwise, $a_i = 0$.
We obtain

$$\begin{aligned} S &= P + av^T \\ G &= \alpha P + (\alpha a + (1-\alpha)e)v^T \end{aligned}$$

Now, the power method applied to G, for any starting vector $x^{(0)T}$

$$\begin{aligned} x^{(k)T} = x^{(k-1)T}G &= \alpha x^{(k-1)T}S + (1-\alpha)x^{(k-1)T}ev^T \\ &= \alpha x^{(k-1)T}S + (1-\alpha)v^T \\ &= \alpha x^{(k-1)T}P + (\alpha x^{(k-1)T}a + (1-\alpha))v^T \end{aligned}$$

where $x^{(k-1)T}e = 1$ because $x^{(k-1)T}$ is a probability vector.
It is easy to see that the power method applied to Google matrix G can be implemented with vector-matrix multiplications on the sparse P, so there is no need to store matrix S or G.
The power method is a matrix-free method that we need considering the size of the Google matrix. But, since P is a sparse matrix, each vector-matrix multiplication in the power method can be computed in $nnz(P)$ flops. Also, this method use minimal storage because at each iteration, the method requires the storage of one vector, the current iterate.

9 Literature

Amy N. Langville, Carl D. Meyer, *Deeper Inside PageRank*, Department of Mathematics, North Carolina State University,Raleigh, NC, July 2004.

Ilse C. F. Ipsen, Teresa Selee, *PageRank Computation, with Special Attention to Dangling Nodes*, SIAM Journal on Matrix Analysis and Applications, January 2007

Z. Vondraček, *Markovljevi lanci*, Zagreb, September 2008.