MAT 167 - PageRank Final Project

Elise Ayala, Jackson Sousa, Kieran Sullivan

August 4, 2024

Abstract

This paper investigates the PageRank algorithm in an effort to understand and explain its purpose and strength. We illustrate the usefulness of the power method as a way to illustrate PageRank's innovation. Lastly, we then give an illustrative example of PageRank in action.

1 Introduction

The PageRank algorithm, developed by Larry Page and Sergey Brin while at Stanford University in the late 1990s, completely revolutionized the ways in which people could navigate the internet. Introduced in their paper, "The Anatomy of a Large-Scale Hypertextual Web Search Engine," PageRank addressed the fundamental challenge of efficiently and effectively surfing through millions of webpages on the internet. In this way, PageRank was not only designed to allow users to index the internet more efficiently, but also to produce more relevant search results than past systems (Brin and Page, 1998). PageRank eventually became the cornerstone of Google's search engine, allowing it to transform into the most popular and effective search tool globally.

2 Methodology

In general, the way PageRank works is by ranking web pages based on their relative importance score pertaining to specific queries. This ranking is based on the idea that the number of total ingoing and outgoing links for a specific webpage gives important information about the relevance of that webpage. Previously, it was often the case that users could get stuck on a webpage that has no outgoing links. This would cause issues as it made surfing the internet inefficient and tedious. In order to illustrate PageRank, a random walk interpretation is helpful in understanding the PageRank algorithm. A random walk is "a process for determining the probable location of a point subject to random motions, given the probabilities (the same at each step) of moving some distance in some direction" (Britannica).

To ameliorate the issue of getting stuck on a webpage, PageRank uses a random walk which induces a Markov chain, a random process in which the next state of a system is determined from the present state of the system alone (Eldén, 2019). So, if a webpage has no outgoing links, it is important to modify the model so that zero rows do not exist. A page that has more ingoing and outgoing links is likely to be more relevant to a search. PageRank then uses an iterative process to rank pages based on its links in a way that mimics at Markov chain since the rank of an individual

page is determined by the ranks of the pages that link to it (Eldén). Overall, this provides a more efficient and effective way to surf the internet.

3 Data Description and Task

Table 1: Webpage Ranks in Decreasing Order

Webpage	Terms
A	Ash, Butternut, Cherry, Elm, Katsura, Magnolia, Teak, Ginkgo
B	Butternut, Fir, Hickory, Magnolia, Pine, Willow, Redwood, Sassafras
C	Ash, Elm, Hickory, Katsura, Oak, Ginkgo, Redwood
D	Butternut, Cherry, Fir, Spruce, Teak, Aspen, Sassafras
E	Cherry, Hickory, Oak, Pine, Willow, Redwood
F	Ash, Fir, Magnolia, Spruce, Ginkgo, Redwood, Aspen, Sassafras
G	Ash, Butternut, Oak, Spruce, Ginkgo, Redwood
H	Ash, Cherry, Hickory, Willow, Redwood, Aspen
I	Elm, Fir, Katsura, Magnolia, Pine, Spruce, Sassafras
J	Magnolia, Oak, Willow, Redwood, Aspen, Sassafras
K	Cherry, Elm, Fir, Hickory, Teak, Ginkgo, Redwood, Sassafras
L	Butternut, Elm, Katsura, Oak, Pine, Spruce, Teak, Ginkgo, Aspen, Sassafras

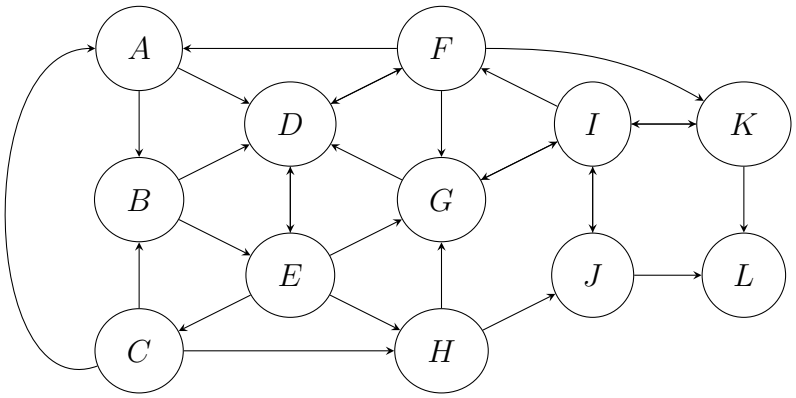

Figure 1: Network Graph

As described previously, the purpose of this project is to illustrate how the PageRank algorithm works. Put simply, the PageRank algorithm finds an eigenvalue of a matrix and its corresponding eigenvector, then uses this information to rank webpages based on relevancy.

We were given 12 webpages listed A through L and their corresponding terms shown in Table 1. We also received a specific Google network to use for this report as shown in Figure 1. The specific queries for this project are shown below.

1. If a user searches for "Ginkgo", what webpage(s) should be returned and in what order?
2. If a user searches for "Hickory" OR "Sassafras" (only one is needed), what webpage(s) should be returned and in what order?

3. If a user searches for "Oak" AND "Pine" (both needed), what webpage(s) should be returned and in what order?

4. If a user searches for "Elm" NOT "Fir" (must have the first, but must NOT have the second), what webpage(s) should be returned and in what order?

The rest of the report shows an example of how PageRank works and some of its properties.

4 Problems and Main Results

4.1

$$G = \begin{bmatrix} 0 & 1/2 & 0 & 1/2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/2 & 1/2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1/3 & 1/3 & 0 & 0 & 0 & 0 & 0 & 1/3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/2 & 1/2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/4 & 1/4 & 0 & 0 & 1/4 & 1/4 & 0 & 0 & 0 & 0 \\ 1/4 & 0 & 0 & 1/4 & 0 & 0 & 1/4 & 0 & 0 & 0 & 1/4 & 0 \\ 0 & 0 & 0 & 1/2 & 0 & 0 & 0 & 0 & 1/2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1/2 & 0 & 0 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1/4 & 1/4 & 0 & 0 & 1/4 & 1/4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/2 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/2 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Figure 2: Raw Google Matrix

Figure 1 is the network graph used for this section of the paper. It can be seen that at nodes have outgoing links, except for node L. A node with no outgoing links is called a dangling node. This can be an issue because it may cause users to get stuck on a webpage by being at a "dead-end."

A raw matrix in Figure 2 was created based on Figure 1. This was done by assigning each row to correspond to a node in alphabetical order, then assigning individual values based on the number of outgoing links per node. For example, for a given node, an element of the row was assigned a value by if it received an outgoing link from the given node and the values of the row were determined by dividing 1 by the the total number of outgoing links. Specifically, node A goes to node B and node D, so the second and fourth element of row 1 receive a value of 1/2. This process was for the entire network to construct the G matrix.

Much like from Figure 1, Figure 2 illustrates that node L is dangling because it is a row that sums to 0. Moreover, it can be seen that all of the other rows sum to 1. This dangling node issue is addressed later.

4.2

$$eig(G^T) = \begin{bmatrix} 0 \\ 0.9324 \\ -0.0930 + .4918i \\ -0.0930 - 0.4918i \\ -0.4835 + 0.2058i \\ -0.4835 - 0.2058i \\ 0.4388 \\ -0.3833 \\ -0.2176 \\ 0.2374 \\ 0.1452 \\ 0 \end{bmatrix}$$

Figure 3: Eigenvalues for G^T

In order to proceed, we must find the eigenvalues of G^T. We use MATLAB to solve $G^T\boldsymbol{\pi} = \boldsymbol{\pi}$. Ideally, one of the eigenvalues would equal 1 in order to represent the steady-state distribution of a Markov chain (Math LibreTexts). However, as seen in Figure 3, none of the eigenvalues of G^T equal 1. This problem is solved later on.

4.3

$$\boldsymbol{\pi} = \begin{bmatrix} -0.1359 \\ -0.1050 \\ -0.0898 \\ -0.5196 \\ -0.3350 \\ -0.3869 \\ -0.3672 \\ -0.1219 \\ -0.4037 \\ -0.1736 \\ -0.2120 \\ -0.2068 \end{bmatrix}$$

Figure 4: Eigenvector Corresponding to λ_2

From Figure 3, it can be seen that none of the eigenvalues of G equal 1. So, we choose the largest eigenvalue which is $\lambda_2 = .9324$ and its corresponding eigenvector called $\boldsymbol{\pi}$. Figure 4 shows that all entries for $\boldsymbol{\pi}$ are negative, so we can simply flip the sign of $\boldsymbol{\pi}$ while keeping the eigenvector equivalent. Changing the sign of $\boldsymbol{\pi}$ is not needed, but is helpful because it improves interpretability of eigenvector results. Moving forward, $\boldsymbol{\pi}$ has the same magnitude for each element, but positive signs.

4.4

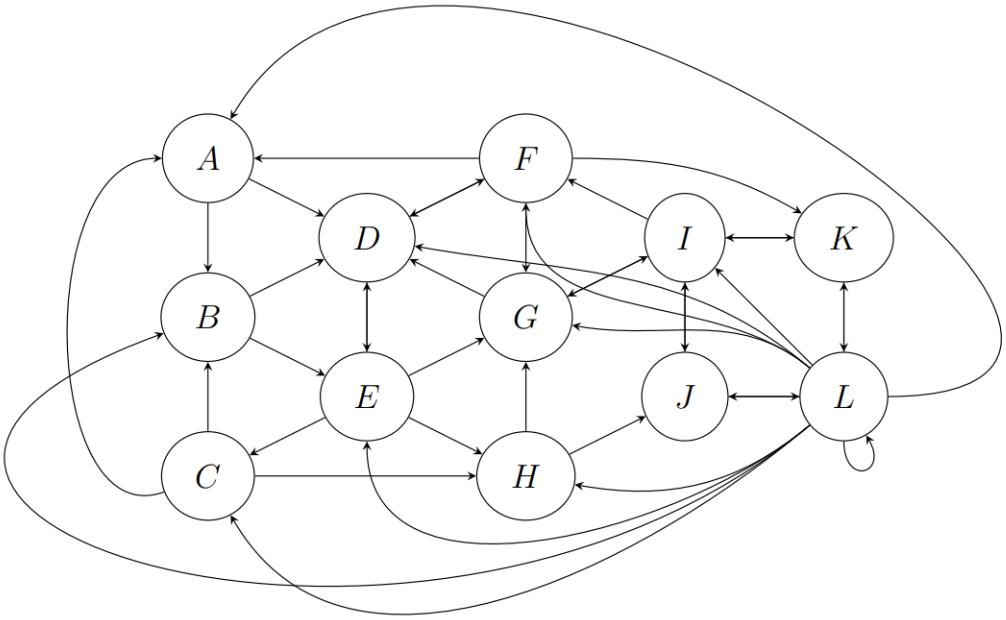

Figure 5: Updated Network Graph With no Dangling Nodes

$$G = \begin{bmatrix} 0 & 1/2 & 0 & 1/2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/2 & 1/2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1/3 & 1/3 & 0 & 0 & 0 & 0 & 0 & 1/3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/2 & 1/2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/4 & 1/4 & 0 & 0 & 1/4 & 1/4 & 0 & 0 & 0 & 0 \\ 1/4 & 0 & 0 & 1/4 & 0 & 0 & 1/4 & 0 & 0 & 0 & 1/4 & 0 \\ 0 & 0 & 0 & 1/2 & 0 & 0 & 0 & 0 & 1/2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1/2 & 0 & 0 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1/4 & 1/4 & 0 & 0 & 1/4 & 1/4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/2 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/2 & 0 & 0 & 1/2 \\ 1/12 & 1/12 & 1/12 & 1/12 & 1/12 & 1/12 & 1/12 & 1/12 & 1/12 & 1/12 & 1/12 & 1/12 \end{bmatrix}$$

Figure 6: Updated Raw Google Matrix

By adding an outgoing links to node L, it is no longer dangling. While not necessary, node L now has outgoing to links to every other node. Indeed, it is clear to see that each row of G now sums to 1. This means that G is now a row stochastic matrix. This is helpful because it ensures an easy probabilistic interpretation since each element of a given row corresponds to a probability. Further, notice that it is now possible to go from any given node to another now that node L is no longer dangling. This means that matrix G is now strongly connected.

For the sake of simplicity and clarity, any time that G is referenced moving forward refers to this updated version of G not the G from part 1.

4.5

$$\text{eig}(G^T) = \begin{bmatrix} 1 \\ -0.0982 + 0.4967i \\ -0.0982 - 0.4967i \\ -0.4884 + 0.2050i \\ -0.4884 - 0.2050i \\ 0.4355 \\ 0.2694 + 0.1427i \\ 0.2694 - 0.1427i \\ -0.3852 \\ -0.2251 \\ 0 \\ -0.2251 \end{bmatrix} \quad \boldsymbol{\pi}_1 = \begin{bmatrix} 0.1457 \\ 0.1250 \\ 0.1023 \\ 0.5131 \\ 0.3371 \\ 0.3742 \\ 0.3637 \\ 0.1364 \\ 0.3984 \\ 0.1859 \\ 0.2112 \\ 0.2166 \end{bmatrix}$$

Figure 7: New Eigenvalues for G^T and the Corresponding Eigenvector for λ_1 Called $\boldsymbol{\pi}_1$

From Figure 7, we can see that unlike before, we have now achieved an eigenvalue of 1. This difference comes because we have altered G to make it a row stochastic matrix. Moreover, it can be seen that all the entries for $\boldsymbol{\pi}_1$ are positive now.

4.6

$\left\|\boldsymbol{\pi}_1\right\|_1 = \boldsymbol{\pi}^T \mathbb{1}_n = \sum_{i=1}^{n} \boldsymbol{\pi}_i = 3.1097$

From the above, we have a 1-norm of 3.1097 for $\boldsymbol{\pi}_1$. While this finding is not incorrect, it is helpful to work with a normalized eigenvector for numerical stability as well as better interepretability of the results. Further explanation for this comes in the next part. To normalize a vector, we divide each component of the vector by the vector's norm as shown below.

$\left\|\boldsymbol{\pi}_1\right\|_1 = \frac{\boldsymbol{\pi_1}}{||\boldsymbol{\pi_1}||} = 1$

This work shows that normalizing the eigenvector has worked since the 1-norm now equals 1. Importantly, other norms such as the 2-norm can be used for this process, but we use the 1-norm for simplicity. Figure 8 below shows the normalized version of $\boldsymbol{\pi}_1$, which we will continue to refer to as $\boldsymbol{\pi}_1$ for simplicity.

$$\boldsymbol{\pi}_1 = \begin{bmatrix} 0.0469 \\ 0.0402 \\ 0.0329 \\ 0.1650 \\ 0.1084 \\ 0.1203 \\ 0.1170 \\ 0.0439 \\ 0.1281 \\ 0.0598 \\ 0.0679 \\ 0.0697 \end{bmatrix}$$

Figure 8: Normalized $\boldsymbol{\pi}_1$

4.7

Table 2: Webpage Ranks of G in Decreasing Order

D	I	F	G	E	L	K	J	A	H	B	C
.1650	.1281	.1203	.1170	.1084	.0697	.0679	.0598	.0469	.0439	.0402	.0329

Each component of $\boldsymbol{\pi}_1$ in Figure 8 corresponds to a ranking score of a given webpage in alphabetical order where a higher score means a higher importance. So the first entry corresponds to webpage A, the second entry corresponds to webpage B, and so on. We have sorted the ranking scores in descending order as to clearly indicate the relative importance of each webpage. We can see that webpages D, I, and F are the most important in this network while webpages H, B, C are the least important in this network. Notably, summing the ranking score of all the components equals 1. This makes sense since we previously normalized $\boldsymbol{\pi}_1$ in the previous part and the ranking scores come from the components of $\boldsymbol{\pi}_1$. The components summing to 1 makes interpreting the ranking score better since the relative importance of one webpage to another can be easily compared.

4.8

When G is a large matrix, it can become computationally challenging to compute the eigenvalues directly. Particularly, methods for computing eigenvalues of a large dense matrix often rely on applying orthogonal transformations to them, which is not possible to do in sparse matrices which are often encountered in PageRank settings (Eldén). Thus, the "power method" is an effective alternative to dealing with this issue by efficiently identifying the dominant eigenvalue and its corresponding eigenvector.

The power method works by taking an initial guess at an eigenvector, usually $\boldsymbol{\pi}_0 = \frac{1}{n}\mathbb{1}_n$ so that the eigenvector is normalized. By construction, we typically have that $1 = \lambda_1 > |\lambda_2| \geq ... \geq |\lambda_n|$ (Eldén). Further, it is possible to represent a vector as a linear combination of eigenvectors. With these ideas in mind, the following math explains the intuition of the power method.

$$
\begin{aligned}
\boldsymbol{\pi}_0 &= c_1 v_1 + c_2 v_2 + ... + c_n v_n \\
G\boldsymbol{\pi}_0 &= c_1 \lambda_1 v_1 + c_2 \lambda_2 v_2 + ... + c_n \lambda_n v_n \\
G^2\boldsymbol{\pi}_0 &= c_1 \lambda_1^2 v_1 + c_2 \lambda_2^2 v_2 + ... + c_n \lambda_n^2 v_n \\
G^j\boldsymbol{\pi}_0 &= c_1 \lambda_1^j v_1 + c_2 \lambda_2^j v_2 + ... + c_n \lambda_n^j v_n \\
&= \lambda_1^j (c_1 v_1 + c_2 (\frac{\lambda_2}{\lambda_1})^j v_2 + ... + c_n (\frac{\lambda_n}{\lambda_1})^j v_n
\end{aligned}
$$

Since $1 = \lambda_1 > |\lambda_i|$, the other terms tend towards 0 as k increases. Thus, the iterative process eventually converges to the eigenvector $\boldsymbol{\pi}_0 = v_1$, where v_1 is the desired eigenvector. The convergence rate of the process is determined by $|\lambda_2|$ in which the closer $|\lambda_2|$ is to 1 means the slower the convergence. This makes sense intuitively since if $|\lambda_2| = .99$, then $(\frac{.99}{1})^j$ would take many iterations to reach 0. Notably, $\lambda_1 = 1$ by construction. However, if this is not the case, then the convergence rate is explicitly determined by $\frac{|\lambda_2|}{\lambda_1}$ instead of $|\lambda_2|$ which implicitly ignores λ_1. The following work illustrates the power method in action. We set $\epsilon = 10^{-6}$, which means that once $||\boldsymbol{\pi}_j - \boldsymbol{\pi}_{j-1}||_1$ less than the threshold ϵ, then stop the power method process.

$$
\boldsymbol{\pi}_0^T = \begin{bmatrix} 1/12 & 1/12 & 1/12 & 1/12 & 1/12 & 1/12 & 1/12 & 1/12 & 1/12 & 1/12 & 1/12 & 1/12 \end{bmatrix}
$$

Table 3: Iteration Number and 1-norm of Eigenvectors

1	2	3	4	5	6	7	8	9	10
.361111	.223379	.112847	.043073	.023361	.010522	.006397	.003585	.001746	.000946
11	12	13	14	15	16	17	18	19	20
0.000355	0.000179	.000096	.000058	.000032	.000016	.000008	.000003	.000002	.0000009

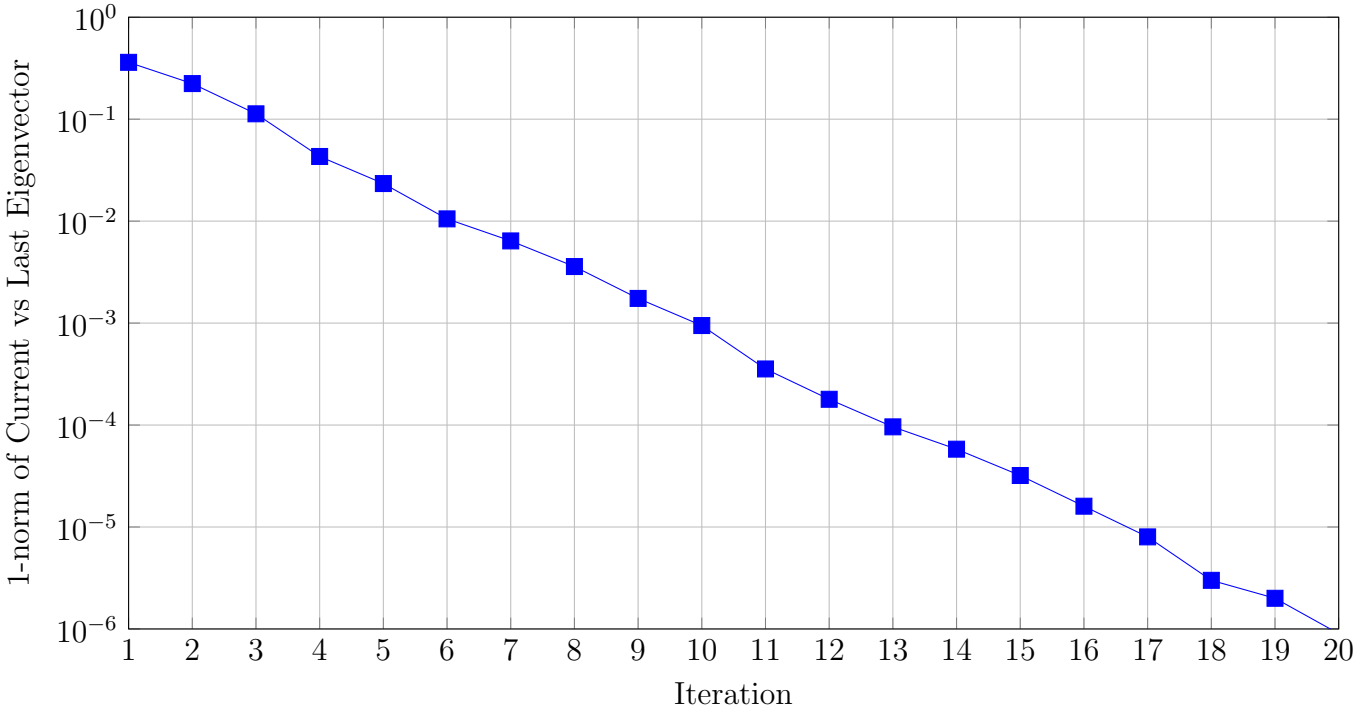

Figure 9: Plot 1-norm of Eigenvectors Over Iterations

After 20 iterations, the power method process stops after reaching the ϵ threshold as shown in Figure 5. Interestingly, the relationship seems close to linear. We find that $\boldsymbol{\pi}_{20}^T$ gives the same results as part 7. This makes sense intuitively since both methods find an eigenvector in mathematically equivalent ways, except with different processes. Below is the result of the power method ranked in descending order of importance with each webpage having the same ranking as Table 2.

$$\boldsymbol{\pi}_{20}^T = \begin{bmatrix} .1650 & .1281 & .1203 & .1170 & .1084 & .0697 & .0679 & .0598 & .0469 & .0439 & .0402 & .0329 \end{bmatrix}$$

4.9

$$\widetilde{G} = \begin{bmatrix} .0125 & .4375 & .0125 & .4375 & .0125 & .0125 & .0125 & .0125 & .0125 & .0125 & .0125 & .0125 \\ .0125 & .0125 & .0125 & .4375 & .4375 & .0125 & .0125 & .0125 & .0125 & .0125 & .0125 & .0125 \\ .2958 & .2958 & .0125 & .0125 & .0125 & .0125 & .0125 & .2958 & .0125 & .0125 & .0125 & .0125 \\ .0125 & .0125 & .0125 & .0125 & .4375 & .4375 & .0125 & .0125 & .0125 & .0125 & .0125 & .0125 \\ .0125 & .0125 & .2250 & .2250 & .0125 & .0125 & .2250 & .2250 & .0125 & .0125 & .0125 & .0125 \\ .2250 & .0125 & .0125 & .2250 & .0125 & .0125 & .2250 & .0125 & .0125 & .0125 & .2250 & .0125 \\ .0125 & .0125 & .0125 & .4375 & .0125 & .0125 & .0125 & .0125 & .4375 & .0125 & .0125 & .0125 \\ .0125 & .0125 & .0125 & .0125 & .0125 & .0125 & .4375 & .0125 & .0125 & .4375 & .0125 & .0125 \\ .0125 & .0125 & .0125 & .0125 & .0125 & .2250 & .2250 & .0125 & .0125 & .2250 & .2250 & .0125 \\ .0125 & .0125 & .0125 & .0125 & .0125 & .0125 & .0125 & .0125 & .4375 & .0125 & .0125 & .4375 \\ .0125 & .0125 & .0125 & .0125 & .0125 & .0125 & .0125 & .0125 & .4375 & .0125 & .0125 & .4375 \\ .0833 & .0833 & .0833 & .0833 & .0833 & .0833 & .0833 & .0833 & .0833 & .0833 & .0833 & .0833 \end{bmatrix}$$

Figure 10: Matrix $\widetilde{G}$ to Ensure that the Matrix is Strongly Conneceted

While our network is already strongly connected as explained earlier, it may be the case that a given network is not strongly connected. This issue can be fixed by constructing $\widetilde{G}$ in the following way and as depicted in Figure 10.

$\alpha = .85$
$E = \frac{1}{n}\mathbb{1}_n\mathbb{1}_n^T$
$\widetilde{G} = \alpha G + (1-\alpha)E = .85G + (1-.85)E = .85G + .15E$

This is equivalent to a user typing in a webpage instead of following a hyper link.

4.10

$$\text{eig}(\widetilde{G}^T) = \begin{bmatrix} 1 \\ -0.0835 + 0.4222i \\ -0.0835 - 0.4222i \\ -0.4151 + 0.1742i \\ -0.4151 - 0.1742i \\ 0.3702 \\ 0.2290 + 0.1213i \\ 0.2290 - 0.1213i \\ -0.3274 \\ -0.1914 \\ 0 \\ -0.0913 \end{bmatrix} \quad \widetilde{\boldsymbol{\pi}}_1 = \begin{bmatrix} 0.1675 \\ 0.1644 \\ 0.1284 \\ 0.4946 \\ 0.3369 \\ 0.3493 \\ 0.3550 \\ 0.1648 \\ 0.3873 \\ 0.2092 \\ 0.2134 \\ 0.2364 \end{bmatrix}$$

Figure 11: New eigenvalues for $\widetilde{G}^T$ and the corresponding eigenvector for $\widetilde{\lambda}_1$ called $\widetilde{\boldsymbol{\pi}}_1$

From Figure 11, we can see that we have an eigenvalue of 1. So, that becomes the dominant eigenvalue with a corresponding eigenvector $\widetilde{\boldsymbol{\pi}}_1$. The corresponding eigenvector $\widetilde{\boldsymbol{\pi}}_1$ originally had all negative entries, but we switched the sign to create all positive entries as down in previous parts.

4.11

Table 4: Webpage Ranks for $\widetilde{G}$ in Decreasing Order

D	I	G	F	E	L	K	J	A	H	B	C
.1542	.1208	.1107	.1089	.1050	.0737	.0665	.0652	.0522	.0514	.0513	.0400

Like before, we normalize the 1-norm of $\widetilde{\boldsymbol{\pi}}_1$ so that $||\widetilde{\boldsymbol{\pi}}_1|| = 1$.

We can see that webpages D, I, and G are the most important in this network while webpages H, B, C are the least important in this network. The overall rankings for $\widetilde{G}$ are very similar to the rankings found for G in part 7 in Table 1. In fact, the only difference in ordering is that webpage G and F are switched. Further, there are differences in the value of each webpage between rankings, but the difference is small. This is confirmed by the fact that $||\boldsymbol{\pi}_1 - \widetilde{\boldsymbol{\pi}}_1|| = .081$.

4.12

$$T = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

Figure 12: Term Document Matrix

For this project, we were assigned specific terms that appear in each webpage and told to do queries for specific terms as outlined by the Data Description and Task section. So, we construct a matrix T that is created from the terms and documents list. Each row of T corresponds to the terms based on the order in which the term first appears. For example, row 1 corresponds to "Ash" and indicates that it appears in webpages A, C, F, G, and H. Similarly, row 2 corresponds to "Butternut" and it appears in A, B, D, G, and L. The following shows the terms that correspond to each row from top to bottom:

Ash, Butternut, Cherry, Elm, Katsura, Magnolia, Teak, Ginkgo, Fir, Hickory, Pine, Willow, Redwood, Sassafras, Oak, Spruce, Aspen.

Notably, there a variety of orderings which could be chosen including alphabetical ordering. The overall findings will not change based on the ordering, so long as the ordering is consistent.

4.13

$$\mathbf{q}_1^T = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

For query 1, we were asked to find which webpages has the word "Gingko." Gingko was the eighth term to appear. So we create a query vector called $\mathbf{q}_1^T$ which has a 1 at the eighth element and 0 for the other elements. By multiplying the $\mathbf{q}_1^T$ with matrix T, the resulting vector would show which webpages contain the word Gingko.

$$\mathbf{q}_2^T = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

For query 2, we were asked to find which webpages has the word "Hickory" or "Sassafras." Hickory was the tenth term to appear and Sassafras was the fourteenth term to appear. So we create a

query vector called $\mathbf{q}_2^T$ which has a 1 at the tenth and fourteenth elements and 0 for the other elements.

$$\mathbf{q}_3^T = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

For query 3, we were asked to find which webpages has the word "Oak" or "Pine." Oak was the eleventh term to appear and Pine was the fifteenth term to appear. So we create a query vector called $\mathbf{q}_3^T$ which has a 1 at the eleventh and fifteenth elements and 0 for the other elements. Each of Oak and Pine will create a positive value when computing $\mathbf{q}_3^T T$. Entries with a value of 2 indicate the presence of Oak and Pine. Since we are only trying to find where Oak or Pine are present, values of 2 are brought down to 1 for simplicity.

$$\mathbf{q}_4^T = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

For query 4, we were asked to find which webpages has the word "Elm" but not "Fir." Elm was the fourth term to appear and Fir was the ninth term to appear. So we create a query vector called $\mathbf{q}_4^T$ which has a 1 at the fourth element, a -1 at the ninth element, and 0 for the other elements. The idea behind this process is that when we computing $\mathbf{q}_4^T T$, negative values will indicate the presence of Fir. So, we turn negative values to a 0.

4.14

As stated previously, if we compute $\mathbf{q}_i^T T$, we will find which webpages fulfill our individual queries. So, we defined $\mathbf{d}_i^T$ to be the vector that satisfies the queries.

$$\mathbf{d}_1^T = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

$\mathbf{d}_1^T$ reveals that Gingko appeared in pages A, C, F, G, K, and L.

$$\mathbf{d}_2^T = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$\mathbf{d}_2^T$ reveals that Hickory or Sassafras appeared in pages B, C, D, E, F, H, I, J, K, and L.

$$\mathbf{d}_3^T = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$\mathbf{d}_3^T$ reveals that Oak and Pine appeared in pages E and L.

$$\mathbf{d}_4^T = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$\mathbf{d}_4^T$ reveals that Elm and not Fir appeared in pages A, C, and L.

4.15

Table 5: Webpage Ranks for Finding Each Term in Decreasing Order

Ginkgo	F	G	L	K	A	C				
Hickory or Sassafras	D	I	F	E	L	K	J	H	B	C
Oak and Pine	E	L								
Elm not Fir	L	A	C							

For ranking the responsive webpages in order of importance, we used the same ranking found in Table 1. The above findings make sense visually since the relative rankings are about the same as the way Table 1 ranked each webpage. For example, F was the third highest ranked webpage from Table 1, and it is highest for Ginkgo and third highest for Hickory or Sassafras. A similar trend follows for other terms. Further, Oak and Pine only have 2 ranked pages since they only appear together twice. Lastly, Elm not Fir only appears 3 times since Elm appears 5 times total, but twice with Fir on webpages I and K.

5 Conclusion

In conclusion, the steps and explanation in this report illustrate the power and importance of the PageRank algorithm. Its efficiency and effectiveness has had significant impact on people's ability to surf and retrieve information from the internet. The mathematics behind PageRank helps users retrieve the most relevant information when searching the internet by providing webpages with the highest ranking score. While the PageRank algorithm is old, it will continue to remain essential to ensure the quality of search results for years to come.

6 Contributions

6.1 Writing Code

Jackson and Kieran focused on coding questions 1 through 11 while Elise focused on questions 12 through 15.

6.2 Writing the Presentation

All 3 members contributed equally to writing the report.

6.3 Tables, Figures, and Graphs

Jackson made the tables, figures, and graphs.

6.4 Assembly of Presentation

All 3 members contributed equally to assembling the presentation.

References

[1] Brin, Sergey, and Lawrence Page. "The anatomy of a large-scale hypertextual web search engine." Computer networks and ISDN systems 30.1-7 (1998): 107-117.

[2] Eldén, Lars. Matrix methods in data mining and pattern recognition. Society for Industrial and Applied Mathematics, 2019.

[3] "Random Walk." Encyclopædia Britannica, Encyclopædia Britannica, inc., 28 June 2024, www.britannica.com/science/random-walk.

[4] "4.5: Markov Chain and Google's PageRank Algorith." Mathematics LibreTexts, Libretexts, 19 June 2024.