Pagerank project

Aebel Shajan

March 2023

Abstract

This report provides an overview of the Pagerank algorithm, its mathematical foundation, and its practical applications. The report is divided into four main sections, each addressing different aspects of the algorithm. The first section provides an introduction to the Pagerank algorithm and its significance. The second section delves into the mathematical foundation of the Pagerank algorithm, including how the power method is utilised. The third section explores the sensitivity of the Pagerank algorithm, examining how changes in the input parameters can affect the final ranking results. Finally, the fourth section presents two case studies demonstrating the practical applications of the Pagerank algorithm. The first example ranks pages linking to the University of Manchester website. The second example looks at an application beyond the web, specifically ranking the imports and exports of EU members.

Contents

1 Introduction **2**

2 Mathematics of Pagerank **3**
2.1 Graph Networks 3
2.2 Mathematical Formulation 3
2.3 Random Walk Interpretation 5
2.4 The Power Method 5
2.5 Stochastic Adjustment 6
2.6 Ensuring a Unique Ranking 7
2.7 The Convergence of the Power Method 10
2.8 Implementation of the Power Method 12
2.9 Rate of Convergence of the Power Method 12

3 Sensitivity of Pagerank **15**

4 Application of the Pagerank Algorithm **17**
4.1 The University of Manchester Website 17
4.2 Imports & Exports of EU Members 20
4.3 Conclusion 21

A MATLAB Code **23**

Chapter 1

Introduction

When searching for information, whether online or in person, we are frequently faced with a multitude of sources. It can be a challenge to sort through these results and find those that are suitable to our needs. To address this issue, Larry Page and Sergey Brin developed the Pagerank algorithm [1], which was famously used to power the Google search engine. The algorithm's central idea is to rank web pages based on their importance, with a page deemed important if it is linked to other important pages. Additionally, the algorithm weights the importance gained from pages that contain many links less heavily.

The mathematical foundations of the Pagerank algorithm are covered in Chapter 2. To begin, the web is depicted as a directed graph. The graph's adjacency matrix is then used as input to construct a ranking vector as output. The problem can be treated as an eigenvalue equation, and iterative methods are employed to solve it. However, some structures in the web can cause problems when using these methods. As a result, changes to the initial input graph must be made to generate a reliable output for the Pagerank vector.

In Chapters 2 and 3, we look at the performance of the algorithm as well as the influence of different parameters on the final ranking of pages. Finally, in Chapter 4, the Pagerank algorithm is implemented with MATLAB and applied to two different scenarios.

Chapter 2

Mathematics of Pagerank

2.1 Graph Networks

The world wide web can be visualised as one big directed graph. The pages on the web correspond to nodes. Links from one page to another correspond to edges. Representing graphs mathematically allows us to manipulate and investigate their properties. A graph contains nodes and edges, these edges can be directed or undirected. The connection between these nodes and edges can be encoded in a square adjacency matrix, denoted by $\mathbf{A}$. Each dimension in $\mathbf{A}$ reflects a node in the graph. An element A_{ij} is non-zero if there is a directed edge going from node i to node j otherwise A_{ij} is zero. The weight of the edge is reflected by the magnitude of A_{ij}. Figure 2.1 shows a directed graph with 7 nodes. The size of the nodes reflects the Pagerank of each node.

The adjacency matrix of the graph in Figure 2.1 is equal to

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}. \tag{2.1}$$

2.2 Mathematical Formulation

The Pagerank algorithm aims to rank the importance of a page by considering the external pages that link to it. The importance of each page relative to others can be quantified by a scalar between 0 and 1. This is known as the Pagerank, denoted $r(P_i)$, for the $P_{i^{th}}$ page/node. Links from more important pages should be weighted more than links from less important ones. Also, the weight should be inversely proportional to the number of links the external pages have. This can be formulated by summing the weighted Pageranks of neighbouring pages,

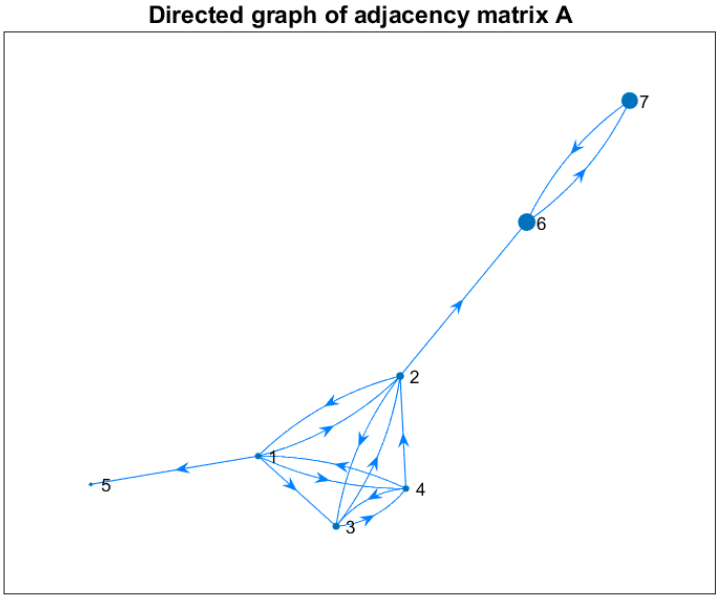

Figure 2.1: Example of directed graph network

P_j, that direct to page P_i :

$$r(P_i) = \sum_{P_j \in B_{P_i}} \frac{r(P_j)}{|P_j|}, \tag{2.2}$$

where $r(P_i)$ is the Pagerank for the ith page and B_{P_i} is the set of all pages j that link to page i. The notation $|P_j|$ refers to the number of links on page P_j. Note that $|P_j|$ is at least 1 because by definition $P_j \in B_{P_j}$ and so must contain a link to page P_i. Let π be known as the page rank vector. The ith element of π is equal to the page rank of page P_i,

$$\pi_i = r(P_i). \tag{2.3}$$

Writing equation (2.2) as a matrix equation using the Pagerank vector yields:

$$\pi = \mathbf{H}\pi. \tag{2.4}$$

Here $\mathbf{H}$ is the column-normalized adjacency matrix of the world wide web. The sum of each column in $\mathbf{H}$ is 1 if there are non-zero elements in that column. Otherwise, the column sum is zero. Adding the extra conditions that $e^T\pi = 1$ and $\pi > 0$ ensures that the rankings are positive and normalised.

If we column normalise the matrix from equation (2.1) we get :

$$\mathbf{H} = \begin{pmatrix} 0 & \frac{1}{3} & 0 & \frac{1}{3} & 0 & 0 & 0 \\ \frac{1}{4} & 0 & \frac{1}{2} & \frac{1}{3} & 0 & 0 & 0 \\ \frac{1}{4} & \frac{1}{3} & 0 & \frac{1}{3} & 0 & 0 & 0 \\ \frac{1}{4} & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 \\ \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{3} & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \tag{2.5}$$

Definition 2.2.1 (Eigenvalue-eigenvector equation)**.** A matrix $S \in \mathbb{R}^{n \times n}$ has eigenvalue $\lambda \in \mathbb{C}^n$ and eigenvector $\nu \in \mathbb{C}$ if the following equation holds true:

$$S\nu = \lambda\nu \text{ with } \nu \neq \mathbf{0}. \tag{2.6}$$

Equation (2.4) is an eigenvalue equation where the Pagerank vector π is the eigenvector of the matrix $\mathbf{H}$ with eigenvalue 1. The matrix $\mathbf{H}$ can have multiple eigenvalues and corresponding eigenvectors. We want to ensure that for all given $\mathbf{H}$ we can find an eigenvector with eigenvalue 1.

Definition 2.2.2 (Dominant eigenvalue)**.** The scalar $\lambda \in \mathbb{C}$ is a dominant eigenvalue of the matrix $\mathbf{S} \in \mathbb{R}^{n \times n}$ if $|\lambda|$ is greater than the modulus of every other eigenvalue of the matrix $\mathbf{S}$.

2.3 Random Walk Interpretation

It is also worth noting that the problem can be stated using the concept of random walks in a Markov chain. Imagine a web surfer who browses the web, page to page, by clicking on a link. The surfer randomly clicks on a link on a page to go to another page. It keeps doing this forever, traversing around the web. The importance of each page would be proportional to how often the surfer visits that page. Markov chains are not used here but the surfer analogy is useful to gain intuition in the steps we take when formulating the Pagerank algorithm.

2.4 The Power Method

Iterative methods can be used to solve for the Pagerank vector in the eigenvalue equation. One such method is the power method; this is the same method Google used early on. The power method is an algorithm that can find the largest eigenvalue and its eigenvector of a given square matrix. The method works by repeatedly multiplying a starting vector by the square matrix. Applying the power method on equation (2.4) gives us :

$$\pi^{(k+1)} = \mathbf{H}\pi^{(k)}. \tag{2.7}$$

There are many problems we face, if we use the power method with matrix $\mathbf{H}$. Firstly we do not know if this iterative process is converging or diverging. Secondly, we want to ensure that it converges only to the Pagerank vector and not some other eigenvector. Some further adjustments on $\mathbf{H}$ before applying the power method.

2.5 Stochastic Adjustment

Definition 2.5.1 (Column Stochastic Matrix)**.** A matrix $S \in \mathbb{R}^{n\times n}$ is column stochastic if all its elements are non-negative and its columns sum to 1:

$$\Sigma_{i=1}^{n}\mathbf{S}_{ij} = 1. \tag{2.8}$$

This can be rewritten in matrix form using the vector $e = [1, ..., 1]^T \in \mathbb{R}^n$:

$$e^T\mathbf{S} = e^T. \tag{2.9}$$

Theorem 1. *Every column stochastic matrix $\boldsymbol{S} \in \mathbb{R}^{n\times n}$ has 1 as an eigenvalue. Moreover, every other eigenvalue of $\boldsymbol{S}$ is less than or equal to 1 in absolute value.*

Proof. The first statement is a direct consequence of the equation (2.9) in the definition of a stochastic matrix. The equation, $e^T\mathbf{S} = e^T$, is an eigenvector equation with e^T as the left eigenvector and 1 as the eigenvalue. The second statement uses the property of the one norm. The general eigenvalue-eigenvector equation of $\mathbf{S}$ can be written as,

$$\mathbf{S}p = \lambda p, \tag{2.10}$$

where $\lambda \in \mathbb{C}$ is any eigenvalue of $\mathbf{S}$ and $p \neq 0 \in \mathbb{R}^n$ is the corresponding eigenvector. If we take the 1 norm of the left-hand side, we get the equation:

$$||\mathbf{S}p||_1 \leq ||\mathbf{S}||_1||p||_1. \tag{2.11}$$

Substitute $\mathbf{S}p = \lambda p$, where λ is an arbitrary eigenvalue of $\mathbf{S}$, and use the fact that the one norm of a stochastic matrix is one:

$$|\lambda|||p||_1 \leq ||p||_1 \tag{2.12}$$

$$|\lambda| \leq 1. \tag{2.13}$$

Hence we can write the eigenvalues of $\mathbf{S}$ as $\{\lambda_1 = 1, \lambda_2, ..., \lambda_n\}$ where each $|\lambda_i| \leq 1$ for $1 \leq i \leq n$. □

So if we know the hyperlink matrix in (2.4) is stochastic then we know that there must exist an eigenvector which solves (2.4) using Theorem 1. In reality, dangling nodes are present in the web and as a result, $\mathbf{H}$ may not be stochastic. When a node in a graph has no outgoing edges, it is known as a "dangling node". To identify such a node, we can check its column in the adjacency matrix and confirm that all the entries are zero. The fifth node in Figure 2.1 is a dangling node. If we look at the adjacency matrix in equation (2.1), we can see that the fifth column is filled with all zeros. On the web, dangling nodes are pages that do not themselves contain links to other pages. Examples include PDFs and images. We adjust matrix $\mathbf{H}$ to account for dangling nodes by adding a new matrix $(\frac{e}{n})a^T$,

$$\mathbf{S} = \mathbf{H} + (\frac{e}{n})a^T, \tag{2.14}$$

where $a \in \mathbb{R}^n$ is a vector representing dangling nodes, with the i^{th} entry equal to 1 if page P_i is a dangling node and 0 otherwise. The vector $e \in \mathbb{R}^n$ has all elements equal to 1.

The addition of $(\frac{e}{n})a^T$ in (2.14) replaces all the 0's in the dangling node column with the value $\frac{1}{n}$. This ensures that the column sums in the resulting matrix $\mathbf{S}$ are all equal to 1, even if there are dangling nodes present in $\mathbf{H}$. The resulting matrix $\mathbf{S}$ is a stochastic matrix, which means it satisfies the condition that all of its elements are non-negative and its columns add up to 1. We can now use Theorem 1 to guarantee that 1 is the largest eigenvalue of $\mathbf{S}$. However, we have no information if the eigenvector associated is unique. Also, we don't know if the eigenvalue 1 is dominant. We require these conditions to ensure that the power method converges and converges only to the Pagerank eigenvector.

We can calculate $\mathbf{S}$ for the graph in Figure 2.1 using its hyperlink matrix $\mathbf{H}$. Notice how the fifth column is no longer empty :

$$\mathbf{S} = \begin{pmatrix} 0 & \frac{1}{3} & 0 & \frac{1}{3} & \frac{1}{7} & 0 & 0 \\ \frac{1}{4} & 0 & \frac{1}{2} & \frac{1}{3} & \frac{1}{7} & 0 & 0 \\ \frac{1}{4} & \frac{1}{3} & 0 & \frac{1}{3} & \frac{1}{7} & 0 & 0 \\ \frac{1}{4} & 0 & \frac{1}{2} & 0 & \frac{1}{7} & 0 & 0 \\ \frac{1}{4} & 0 & 0 & 0 & \frac{1}{7} & 0 & 0 \\ 0 & \frac{1}{3} & 0 & 0 & \frac{1}{7} & 0 & 1 \\ 0 & 0 & 0 & 0 & \frac{1}{7} & 1 & 0 \end{pmatrix}. \tag{2.15}$$

2.6 Ensuring a Unique Ranking

The power method converges to the eigenvector corresponding to the dominant eigenvalue of a given square matrix. More information is provided in Section 2.7. So far we have shown that $\mathbf{S}$ has an eigenvalue of 1 with the modulus of all other eigenvalues being less than or equal to 1. This is not enough to prove that the Power method uniquely converges to the Pagerank vector. As a result, we make further adjustments and construct the Google matrix $\mathbf{G} \in \mathbb{R}^{n \times n}$ using $\mathbf{S}$ from (2.14):

$$\mathbf{G}(\alpha) = \alpha\mathbf{S} + (1-\alpha)\nu e^T. \tag{2.16}$$

Here two new parameters are introduced, α and ν, which determine how a surfer "teleports" in the random walk model. The random walk model consists of a surfer who traverses the web by randomly clicking on links. Now we allow the surfer to teleport instead of clicking a link. The transition probability $\alpha \in (0,1)$ determines how often the surfer teleports. The personalisation vector $\nu \in \mathbb{R}^n$ is a positive probability vector with the property $e^T\nu = 1$. The matrix formed by the personalisation vector νe^T is called the teleportation matrix. This matrix describes which node the random surfer teleports to based on which node it is currently on.

Proposition 1. Let the personalisation vector be positive, $\nu > 0$. Then all the entries in the Google matrix are positive. i.e. $\mathbf{G}(\alpha) > 0$.

Proof. The personalisation vector and vector of ones are known to be positive, $\nu > 0, e > 0$. The matrix defined by νe^T is then also positive as it is the outer product of the two positive vectors. The matrix $\mathbf{S}$ is non-negative as its stochastic. Since $\alpha \in (0,1)$ then $1-\alpha > 0$. The matrix $\mathbf{G}$ is positive because the addition of a non-negative matrix to a positive matrix gives a positive matrix. □

Proposition 2. The Google matrix $\mathbf{G}$ is stochastic.

Proof. The matrix $\mathbf{S} \in \mathbb{R}^{n\times n}$ is already stochastic. The teleportation matrix νe^T is a matrix where all the columns equal the vector ν. As ν is a probability vector, with $e^T\nu = 1$, the column sums of this matrix will be equal to 1, hence the teleportation vector νe^T is stochastic. The column sums of $\alpha\mathbf{S}$ equal α, whilst the column sums of $(1-\alpha)\nu e^T$ equal $1-\alpha > 0$. Adding both matrices together gives another matrix with column sums equal to 1. □

Perron's Theorem is a fundamental result in linear algebra and matrix theory. It can be used to establish the uniqueness and existence of the Pagerank vector for the Google matrix.

Definition 2.6.1 (Spectral Radius)**.** The spectral radius of a matrix is equal to the absolute value of the eigenvalue that is greater than or equal to all other eigenvalues of that matrix. It is denoted by

$$\rho(\mathbf{A}) = \max\{|\lambda| : \lambda \textit{ is an eigenvalue of } \mathbf{A}\}. \tag{2.17}$$

Theorem 2 (Perron's Theorem)**.** *If $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{A} > 0$ then:*

1. *$\rho(\boldsymbol{A}) > 0$.*
2. *$\rho(\boldsymbol{A})$ is an eigenvalue of $\boldsymbol{A}$.*
3. *There is a corresponding eigenvector $x > 0$ with $\boldsymbol{A}x = \rho(\boldsymbol{A})x$.*
4. *$\rho(\boldsymbol{A})$ is an eigenvalue that only appears once as a root of the characteristic polynomial of $\boldsymbol{A}, \det(\boldsymbol{A} - \lambda\boldsymbol{I}) = 0$*
5. *All the other eigenvalues are less than $\rho(\boldsymbol{A})$ in absolute value, i.e., $\rho(\boldsymbol{A})$ is the only eigenvalue of maximum modulus.*

Proof. The proof can be found in [2]. □

Proposition 3. The Google matrix $\mathbf{G} \in \mathbb{R}^{n\times n}$ has a dominant eigenvalue equal to 1.

Proof. By Theorem 1, as $\mathbf{G}$ is stochastic, it has an eigenvalue equal to 1 with the modulus of all other eigenvalues being less than or equal to 1. Hence the spectral radius of $\mathbf{G}$ is equal to 1, $\rho(\mathbf{G}) = 1$. As $\mathbf{G}$ is also positive by Proposition 1, we can use Perron's theorem to deduce that $\rho(\mathbf{G}) = 1$ is the dominant eigenvalue of $\mathbf{G}$. I.e. there is only one eigenvalue equal to 1 and all other eigenvalues are less than or equal to 1. □

Proposition 4. Let $u \in \mathbb{C}^n$ be an eigenvector of a matrix $A \in \mathbb{C}^{n\times n}$ with eigenvalue $\lambda \in \mathbb{C}$. Let $v \in \mathbb{C}^n$ be an eigenvector of A^T with eigenvalue $\mu \in \mathbb{C}$ such that $\lambda \neq \mu$. Then $u^T v = 0$.

Proof. Taking the transpose of $Au = \lambda u$ gives:

$$u^T A^T = \lambda u^T.$$

Multiplying by v from the right gives

$$u^T A^T v = \lambda u^T v. \tag{2.18}$$

As v is an eigenvector of A^T with eigenvalue $\mu \neq \lambda$, we have

$$A^T v = \mu v.$$

Multiplying this equation by u^T from the left gives:

$$u^T A^T v = \mu u^T v. \tag{2.19}$$

If we let the right hand sides of equations (2.18) and (2.19) equal, we get:

$$\lambda u^T v = \mu u^T v,$$

which can only be true if $u^T v = 0$ as $\lambda \neq \mu$. □

Proposition 5. Let $\lambda_2 \in \mathbb{C}$ be the second largest eigenvalue of the Google matrix $\mathbf{G} \in \mathbb{R}^{n \times n}$. Then $|\lambda_2| \leq \alpha$, where $\alpha \in (0, 1)$ is the transition probability defined in equation 2.16.

Proof. The following proof is taken from [3].

First we prove that the following is true:

$$e^T x_2 = 0, \tag{2.20}$$

where $e \in \mathbb{R}^n$ is the vector of all ones and $x_2 \in \mathbb{R}^n$ is the right hand eigenvector corresponding to eigenvalue λ_2 of $\mathbf{G}$. As $\mathbf{G}$ is stochastic, the vector e is an eigenvector of G^T and has eigenvalue equal to 1. The eigenvector x_2 has eigenvalue λ_2 which is less than 1 by Proposition 3. Therefore we can use Proposition 4 to prove that $e^T x_2 = 0$.

Now let's look at the eigenvalue-eigenvector equation for x_2:

$$\lambda_2 x_2 = \mathbf{G} x_2 \tag{2.21}$$

$$= \alpha \mathbf{S} x_2 + (1 - \alpha) \nu e^T x_2. \tag{2.22}$$

As $e^T x_2 = 0$ we have:

$$\lambda_2 x_2 = \alpha \mathbf{S} x_2. \tag{2.23}$$

Hence we have that x_2 is also an eigenvector of the stochastic matrix S with eigenvalue λ_2/α. By Theorem 1, the modulus of the eigenvalues of a stochastic matrix must be less than or equal to 1, therefore:

$$\frac{|\lambda_2|}{|\alpha|} \leq 1 \tag{2.24}$$

$$|\lambda_2| \leq \alpha, \tag{2.25}$$

where we have used $|\alpha| = \alpha$ since $\alpha > 0$. □

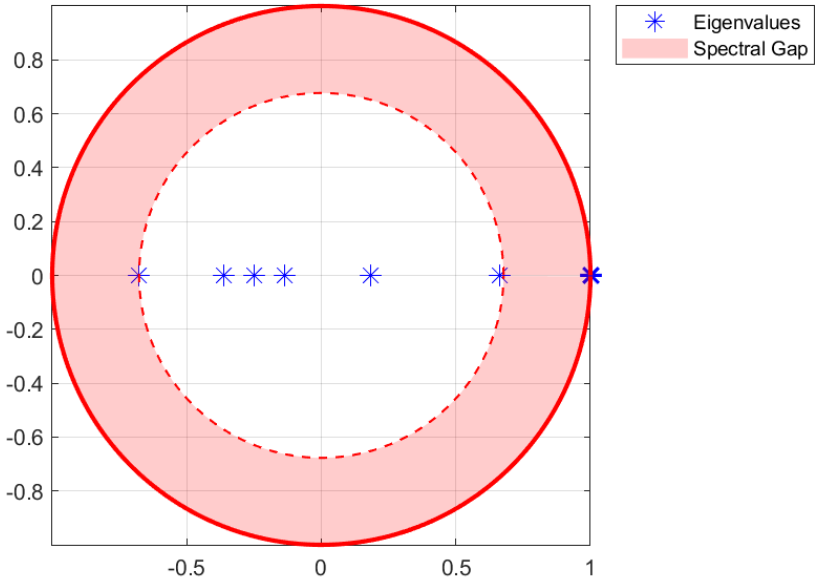

Figure 2.2: Plot of the eigenvalues of $\mathbf{G}$ from (2.26) in the complex plane.

Returning to the example graph in Figure 2.1 we can calculate the corresponding Google matrix with $\alpha = 0.85$ and $\nu = [\frac{1}{7}, \frac{1}{7}, \frac{1}{7}, \frac{1}{7}, \frac{1}{7}, \frac{1}{7}, \frac{1}{7}]^T$:

$$\mathbf{G} = \begin{pmatrix} \frac{3}{140} & \frac{32}{105} & \frac{3}{140} & \frac{32}{105} & \frac{1}{7} & \frac{3}{140} & \frac{3}{140} \\ \frac{131}{560} & \frac{3}{140} & \frac{25}{56} & \frac{32}{105} & \frac{1}{7} & \frac{3}{140} & \frac{3}{140} \\ \frac{131}{560} & \frac{32}{105} & \frac{3}{140} & \frac{32}{105} & \frac{1}{7} & \frac{3}{140} & \frac{3}{140} \\ \frac{131}{560} & \frac{3}{140} & \frac{25}{56} & \frac{3}{140} & \frac{1}{7} & \frac{3}{140} & \frac{3}{140} \\ \frac{131}{560} & \frac{3}{140} & \frac{3}{140} & \frac{3}{140} & \frac{1}{7} & \frac{3}{140} & \frac{3}{140} \\ \frac{3}{140} & \frac{32}{105} & \frac{3}{140} & \frac{3}{140} & \frac{1}{7} & \frac{3}{140} & \frac{61}{70} \\ \frac{3}{140} & \frac{3}{140} & \frac{3}{140} & \frac{3}{140} & \frac{1}{7} & \frac{61}{70} & \frac{3}{140} \end{pmatrix}. \tag{2.26}$$

Notice how all the elements in equation (2.26) are positive.

2.7 The Convergence of the Power Method

We have transformed the initial hyperlink matrix $\mathbf{H}$ by making it both stochastic and positive. This has led to the creation of the Google matrix $\mathbf{G}$. Now that we have obtained $\mathbf{G}$, we can establish the convergence of the power method on $\mathbf{G}$.

Definition 2.7.1 (Rank one matrix)**.** The rank of a matrix is defined as the maximum number of linearly independent columns the matrix has. A rank one matrix only has one linearly independent column.

Proposition 6. A rank one matrix $\mathbf{A}$ can be written as the outer product of two vectors.

Theorem 3 (The powers of positive matrices converge to a rank one matrix.)**.** *If $\boldsymbol{A} > 0$, x is any positive eigenvector of $\boldsymbol{A}$ corresponding to $\rho(\boldsymbol{A})$, and y is any positive eigenvector of $\boldsymbol{A}^T$ corresponding to $\rho(\boldsymbol{A}^T) = \rho(\boldsymbol{A})$ then*

$$\lim_{k\to\infty} \left(\frac{\boldsymbol{A}}{\rho(\boldsymbol{A})}\right)^k = \frac{xy^T}{y^T x}. \tag{2.27}$$

Proof. The proof is provided in [2]. □

Proposition 7. Let $\pi^{(0)} \in \mathbb{R}^n$ and $e = [1, ..., 1]^T$ such that $e^T\pi^{(0)} = 1$. For the Google matrix $\mathbf{G}$ in (2.16), let:

$$\pi^{(k+1)} = \mathbf{G}(\alpha)\pi^{(k)}. \tag{2.28}$$

Then:

$$\lim_{k\to\infty} \pi^{(k)} = \pi, \tag{2.29}$$

where $\pi \in \mathbb{R}^n$ such that:

$$\mathbf{G}\pi = \pi,\ \pi > 0,\ e^T\pi = 1. \tag{2.30}$$

Proof. The Google matrix $\mathbf{G}$ is stochastic so $\rho(\mathbf{G}) = 1$ by Theorem 1 and also $\mathbf{G} > 0$ by Proposition 1. Also, we have that $e^T\mathbf{G} = e^T$, which is the same as saying, $\mathbf{G}^T e = e$. Hence e is a positive eigenvector of $\mathbf{G}$. The matrix $\mathbf{G}$ is positive by Proposition 1. The Pagerank eigenvector π satisfies $\mathbf{G}\pi = \pi$. This means we can apply Theorem 3 to $\mathbf{G}$ with $x = \pi$ and $y = e$ to obtain:

$$\lim_{k\to\infty} \pi^{(k)} = \lim_{k\to\infty} (\mathbf{G})^k\pi^{(0)} = \frac{\pi e^T}{e^T\pi}\pi^{(0)} = \pi. \tag{2.31}$$

□

Using the Google matrix defined in (2.26), the power method was used to find the rankings of each node in Figure 2.1. The Pagerank of each node is shown in Table 2.1.

Pagerank	Node label
0.29381	6
0.27659	7
0.11249	2
0.10131	3
0.087654	4
0.083551	1
0.044599	5

Table 2.1: The Pagerank algorithm was performed using the matrix (2.26) with $\alpha = 0.85$ and $\nu = [\frac{1}{7}, \frac{1}{7}, \frac{1}{7}, \frac{1}{7}, \frac{1}{7}, \frac{1}{7}, \frac{1}{7}]^T$. The Pageranks for each node are listed in descending order as well as the label associated with each node.

2.8 Implementation of the Power Method

The Google matrix $\mathbf{G}$ can become a very large matrix (on the order of billions) if you consider how big the web is. To account for this we can express the dense matrix $\mathbf{G}$ in terms of the very sparse hyperlink matrix $\mathbf{H}$. We can do this by substituting the definition of $\mathbf{S}$ from (2.14) into the definition of $\mathbf{G}$ from (2.16):

$$\mathbf{G}(\alpha) = \alpha\mathbf{H} + \frac{e}{n}(\alpha a^T - \alpha e^T + e^T). \tag{2.32}$$

As a result, we only need to store the normalised adjacency matrix $\mathbf{H}$, the teleportation parameter α, the dangling node vector a^T and the current Pagerank vector $\pi^{(k)}$. Less storage space is required to store these parameters compared to storing the full dense matrix $\mathbf{G}$.

Substituting the equation for $\mathbf{G}$ given by (2.32) into the power method (2.28) gives:

$$\begin{aligned}
\pi^{(k+1)} &= \mathbf{G}(\alpha)\pi^{(k)} && (2.33)\\
&= \alpha\mathbf{H}\pi^{(k)} + \frac{e}{n}(\alpha a^T\pi^{(k)} - \alpha e^T\pi + e^T\pi) && (2.34)\\
&= \alpha\mathbf{H}\pi^{(k)} + \frac{e}{n}(\alpha a^T\pi^{(k)} - \alpha + 1). && (2.35)
\end{aligned}$$

On the third line, we have used $e^T\pi^{(k)} = 1$. This is a direct consequence of starting with $\pi^{(0)} > 0$ such that $e^T\pi^{(0)} = 1$ and having $\mathbf{G}$ such that $e^T G = e^T$. Equation (2.35) implies that only the multiplication of π by $\mathbf{H}$ and a^T need to be considered at each iteration. For the term $\mathbf{H}\pi$, the number of multiplication operations required is equal to the number of non-zeros in $\mathbf{H}$. The term $a^T\pi$ takes at most n multiplication operations. This means that overall less operations per iteration are needed using (2.35) compared to if we had calculated $\mathbf{G}$ then used $\pi^{(k+1)} = \mathbf{G}(\alpha)\pi^{(k)}$.

2.9 Rate of Convergence of the Power Method

Let $\mathbf{A} \in \mathbb{R}^{n\times n}$ be a matrix with eigenvalues $|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq .... \geq |\lambda_n|$ where each $\lambda_i \in \mathbb{C}$. Let $x_1, ..., x_n$ be the corresponding, linearly independent eigenvectors, with each $x_i \in \mathbb{C}^n$. Then any vector $b^{(0)} \in \mathbb{R}^n$ can be expressed in terms of the eigenvectors,

$$b^{(0)} = c_1x_1 + ... + c_nx_n, \tag{2.36}$$

where $c_i \in \mathbb{R}$ is the i^{th} coefficient. Applying the power method k times yields

$$\begin{aligned}
b^{(k)} &= \mathbf{A}^k b^{(0)} && (2.37)\\
&= \sum_{i=1}^{n} \lambda_i^k c_i x_i && (2.38)\\
&= \lambda_1^k (c_1x_1 + \sum_{i=2}^{n} (\frac{\lambda_i}{\lambda_1})^k c_i x_i). && (2.39)
\end{aligned}$$

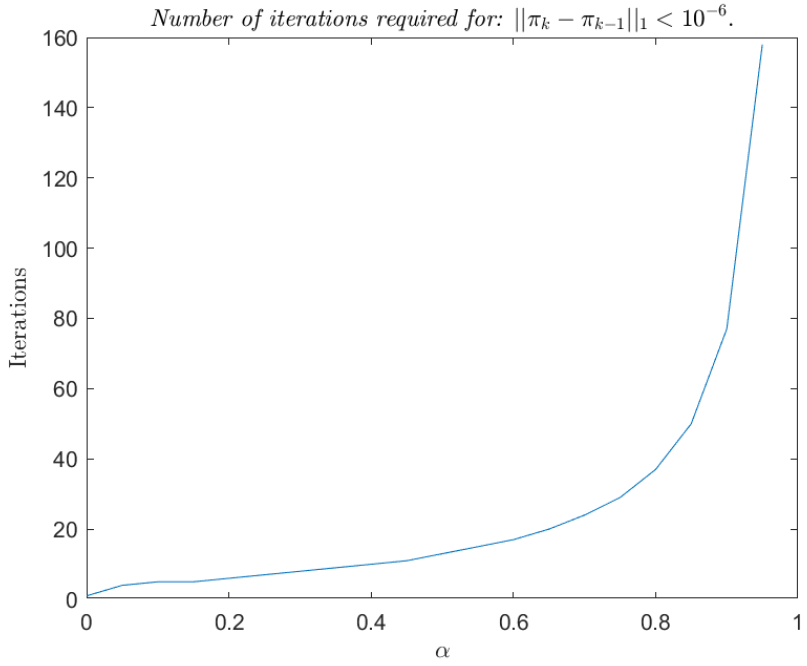

Figure 2.3: The Pagerank algorithm was performed using the matrix (2.26) with varying α and $\nu = [\frac{1}{7}, \frac{1}{7}, \frac{1}{7}, \frac{1}{7}, \frac{1}{7}, \frac{1}{7}, \frac{1}{7}]^T$. The plot shows how the number of iterations for the algorithm converges for different values of α. The criteria for convergence was to have $||\pi^{(k)} - \pi^{(k-1)}||_1 < 10^{-6}$.

As k approaches infinity we have that $b^{(k)}$ approaches the eigenvector x_1 with the dominant eigenvalue λ_1. This is because $(\frac{\lambda_i}{\lambda_1})^k$ goes to zero as k approaches infinity since $\lambda_i < \lambda_1 \ \forall i > 1$. The dominant term is $(\frac{\lambda_2}{\lambda_1})^k$, it is the ratio of the dominant to the subdominant eigenvalue which influences the rate of convergence of the power method. Now let's apply this result to the Google matrix $\mathbf{G}$. By Proposition 5 $\lambda_2 \leq \alpha$ and $\lambda_1 = 1$ by Proposition 3, therefore the rate of convergence of the algorithm for $\mathbf{G}$ is at most α^k. So the smaller the value α, the fewer iterations it takes for α^k to approach zero. Choosing a high value of α requires a greater number of iterations to reach the same accuracy that a lower value of α gives. This relationship is plotted in Figure 2.3.

We can measure the convergence of the power method by calculating the one norm of the difference between each iteration and the actual eigenvector of the Google matrix ($||\pi - \pi^{(k)}||_1$). Another metric that measures convergence is the one norm of the difference between the current and previous iterations. Both of these metrics were recorded during the application of the Pagerank algorithm to the graph in Figure 2.1. The results are plotted in Figure 2.4. Both plots show a roughly linear convergence.

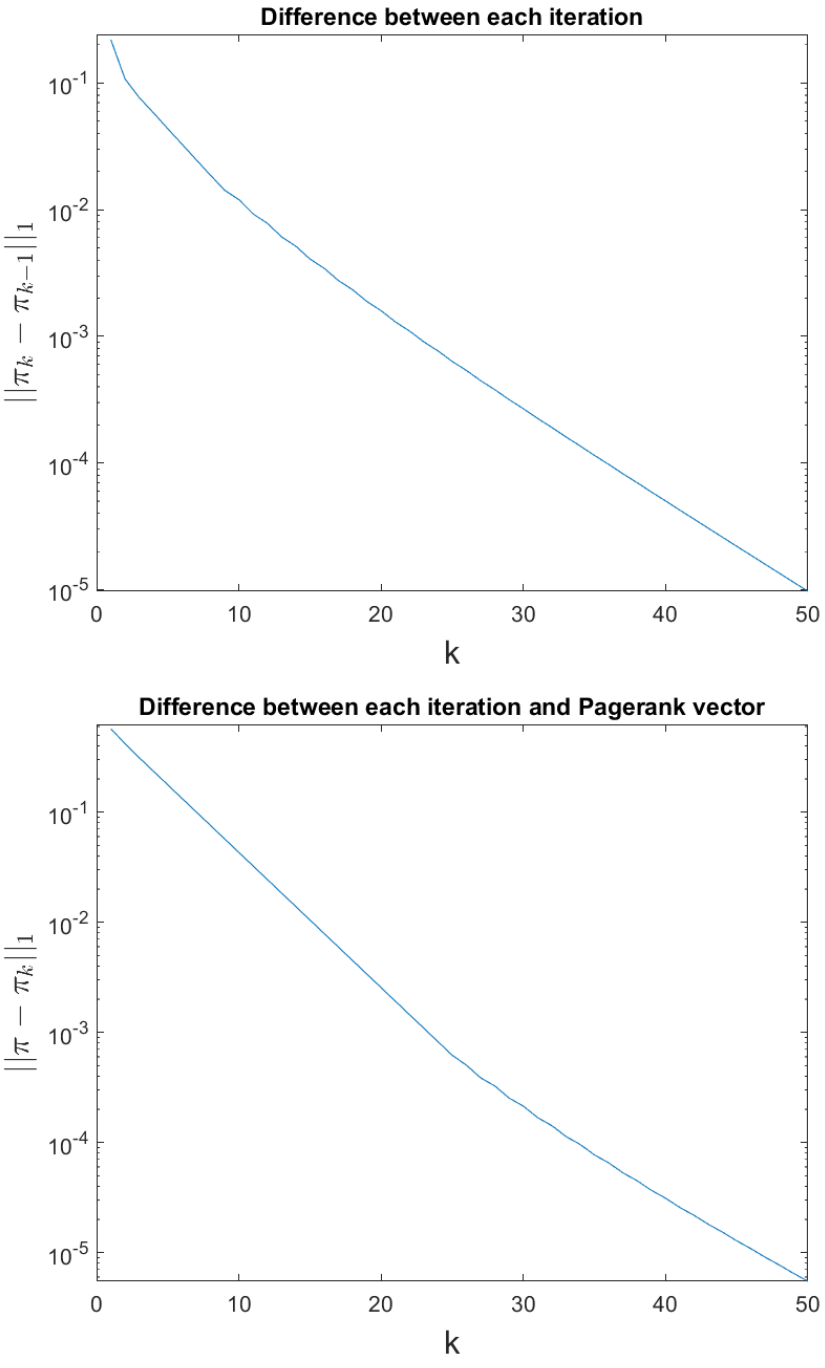

Figure 2.4: The Pagerank algorithm was performed using the matrix (2.26) with $\alpha = 0.85$ and $\nu = [\frac{1}{7}, \frac{1}{7}, \frac{1}{7}, \frac{1}{7}, \frac{1}{7}, \frac{1}{7}, \frac{1}{7}]^T$. Logarithmic plots showing the convergence of the power method at each iteration k.

Chapter 3

Sensitivity of Pagerank

The Google matrix is a function of the transition probability α, the original adjacency matrix $\mathbf{A}$ and the personalisation vector ν. We now discuss the impact of α on the resulting Pagerank vector. Ideally, we should choose the optimum α and ν such that rankings are reliable and can be obtained in a reasonable amount of time.

The parameter α determines how influential the personalisation vector $\nu \in \mathbb{R}^n$ is. It describes how likely the random surfer is to click a link on the page rather than teleport randomly to another page on the web. The derivative of the Pagerank vector with respect to α quantifies this. To derive this derivative, we start off by expressing the Pagerank vector in terms of α and ν:

$$\begin{aligned} \pi &= \mathbf{G}(\alpha)\pi \\ &= \alpha\mathbf{S}\pi + (1-\alpha)\nu e^T\pi. \end{aligned} \tag{3.1}$$

As π represents a probability vector the column sum should be equal to one. i.e. $e^T\pi = 1$. Hence (3.1) can be rewritten as :

$$(\mathbf{I} - \alpha\mathbf{S})\pi = (1-\alpha)\nu. \tag{3.2}$$

Multiplying from the left by $(\mathbf{I} - \alpha\mathbf{S})^{-1}$ yields,

$$\pi(\alpha) = (\mathbf{I} - \alpha\mathbf{S})^{-1}(1-\alpha)\nu. \tag{3.3}$$

We then differentiate this equation with respect to α, using the formula,

$$\frac{d\mathbf{A}^{-1}}{d\alpha} = -\mathbf{A}^{-1}(\frac{d\mathbf{A}}{d\alpha})\mathbf{A}^{-1}. \tag{3.4}$$

This formula and more details on why the Pagerank vector can be differentiated are given in [4]. Applying the above equation to differentiate $(\mathbf{I} - \alpha\mathbf{S})^{-1}$ gives,

$$\begin{aligned} \frac{d\pi}{d\alpha} &= (\mathbf{I} - \alpha\mathbf{S})^{-1}\mathbf{S}(\mathbf{I} - \alpha\mathbf{S})^{-1}(1-\alpha)\nu - (\mathbf{I} - \alpha\mathbf{S})^{-1}\nu \\ &= -[(\mathbf{I} - \alpha\mathbf{S})^{-1}\mathbf{S}(-1+\alpha) + \mathbf{I}](\mathbf{I} - \alpha\mathbf{S})^{-1}\nu \\ &= -(\mathbf{I} - \alpha\mathbf{S})^{-1}[-\mathbf{S} + \alpha\mathbf{S} + \mathbf{I} - \alpha\mathbf{S}](\mathbf{I} - \alpha\mathbf{S})^{-1}\nu \\ &= -(\mathbf{I} - \alpha\mathbf{S})^{-2}(\mathbf{I} - \mathbf{S})\nu, \end{aligned} \tag{3.5}$$

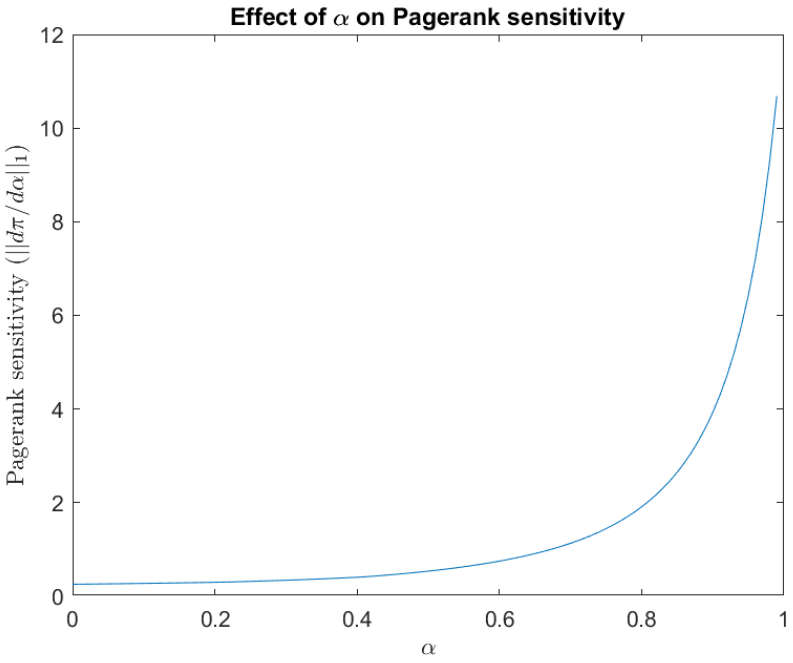

Figure 3.1: The Pagerank algorithm was performed using the matrix (2.26) with varying α and $\nu = [\frac{1}{7}, \frac{1}{7}, \frac{1}{7}, \frac{1}{7}, \frac{1}{7}, \frac{1}{7}, \frac{1}{7}]^T$. The graph shows the effect of the alpha parameter on the Pagerank algorithm's sensitivity.

where in the last step we used the fact that $(\mathbf{I} - \mathbf{S})$ commutes with $(\mathbf{I} - \alpha\mathbf{S})^{-1}$. The proof for this is again given in [5].

Figure 3.1 shows how the Pagerank sensitivity for the graph in Figure 2.1 varies for different values of alpha. There is an exponential increase in the sensitivity as α is increased from 0.5 to 1. As α approaches 1, the sensitivity vector becomes very large. A value of 0.85 is normally used in applications since it is roughly after that point the Pagerank sensitivity increases rapidly.

Chapter 4

Application of the Pagerank Algorithm

The Pagerank algorithm can be applied to many different situations other than ranking websites. The algorithm at its core, ranks nodes in a graph based on how many "important nodes" point to it. If a problem can be formulated as a directed graph then we can use the Pagerank algorithm on it. Examples of applications include:

- Ranking of athletes at a match.
- Ranking of animals in a food web.
- Ranking the best way to distribute medicine.
- Ranking accounts in a social network.

More detail of applications is presented in [5].

The Pagerank algorithm was written in MATLAB and applied to two different scenarios. The MATLAB code for the Pagerank function is in Appendix A. The full source code can be found at [6].

4.1 The University of Manchester Website

A web crawler was used to rank the websites that are linked by `www.manchester.ac.uk`. The code for the web crawler is in Appendix A. The crawler works by first collecting the links on the initial page and appending them to an array corresponding to that page. Then it crawls the pages linked to by the elements in the array to get another array full of links from that page. The depth parameter controls how many times this process repeats. After this, the links are all aggregated into one list where no link appears twice. An adjacency matrix is obtained from the aggregated list by crawling each link in the list and seeing if other elements are on the link's page. The adjacency matrix and the website links are stored in the file system. The adjacency matrix for `www.manchester.ac.uk` is shown in Figure 4.1.

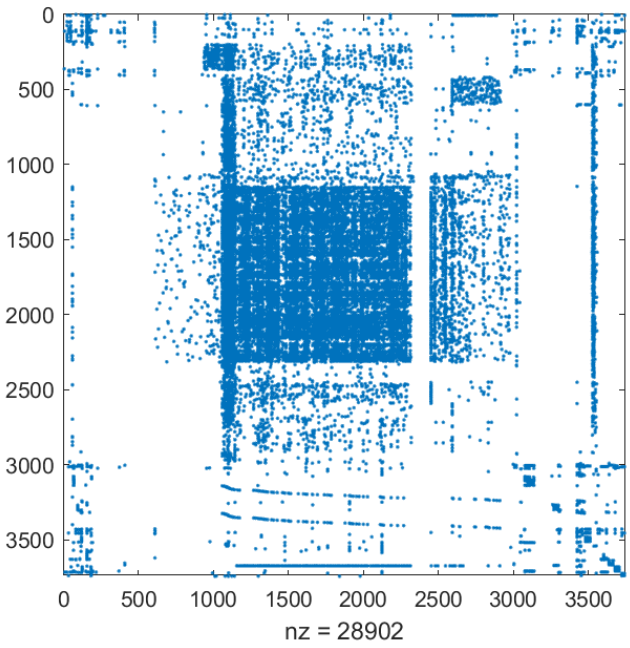

Figure 4.1: Plot showing the sparse adjacency matrix of the websites crawled from `www.manchester.ac.uk`.

The corresponding graph is plotted in Figure 4.2 and the Pageranks are shown in Table 4.2. The massive cluster of nodes near the bottom left of Figure 4.2 all contain `www.research.manchester.ac.uk`. These websites are the profiles of researchers at Manchester University, which explains why there are so many nodes. This also explains why most of the top-ranking websites are research related.

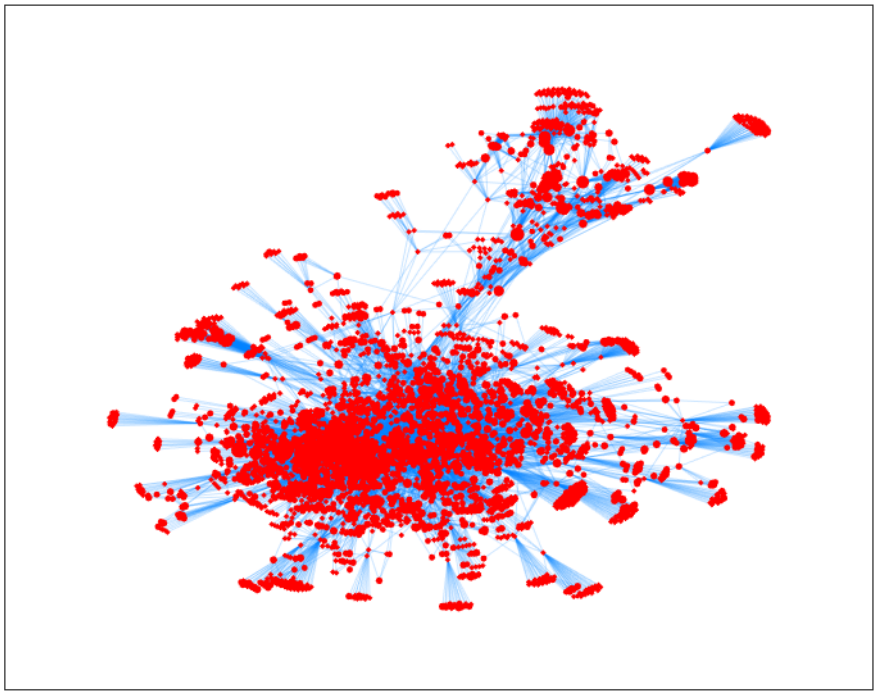

Figure 4.2: Graph of websites that link to and are linked by "www.manchester.ac.uk". The size of the nodes represents their relative Pagerank. The nodes represent websites and the edge show links between them.

Table 4.1: Top 10 websites with highest number of inward links.

Inward Links	Website
3.25×10^2	https://www.un.org/sustainabledevelopment/sustainable-development-goals/
1.15×10^2	https://research.manchester.ac.uk/en/persons/kevin.munro
1.09×10^2	https://research.manchester.ac.uk/en/persons/katherine.payne
1.09×10^2	https://research.manchester.ac.uk/en/persons/tony.heagerty
1.08×10^2	https://research.manchester.ac.uk/en/persons/kim.m.linton
1.07×10^2	https://research.manchester.ac.uk/en/persons/philip.crosbie
1.06×10^2	https://research.manchester.ac.uk/en/persons/emma.crosbie
1.00×10^2	https://research.manchester.ac.uk/en/persons/jill.clayton-smith
9.80×10^1	https://research.manchester.ac.uk/en/persons/andrew.renehan
9.80×10^1	https://research.manchester.ac.uk/en/persons/chris.plack

Table 4.2: Top 10 websites with highest Pagerank.

Pagerank	Website
1.14×10^{-2}	https://research.manchester.ac.uk/en/persons/alex.henderson
1.02×10^{-2}	https://research.manchester.ac.uk/en/persons/herve.boutin
8.84×10^{-3}	https://research.manchester.ac.uk/en/activities/european-society-for-molecular-imaging-esmi-external-organisation-2
8.62×10^{-3}	https://www.un.org/sustainabledevelopment/sustainable-development-goals/
6.53×10^{-3}	http://creativecommons.org/licenses/by/4.0/
6.15×10^{-3}	https://research.manchester.ac.uk/en/persons/yifan.xu
4.32×10^{-3}	https://research.manchester.ac.uk/en/persons/wolodymyr.krywonos
4.00×10^{-3}	https://www.linkedin.com/school/university-of-manchester/
3.77×10^{-3}	https://research.manchester.ac.uk/en/persons/p.dudek
3.76×10^{-3}	https://research.manchester.ac.uk/en/persons/dirk.koch

4.2 Imports & Exports of EU Members

The PageRank algorithm can be used in other applications beyond the web. Here, the algorithm ranks the exports and imports of EU member countries by treating them as nodes in a directed graph. The graph's edges represent the trading relationships between countries, and the weight of each edge represents the volume of trade. The data was extracted from the Eurostat data browser website [7] in the form of an Excel spreadsheet. Each row in the spreadsheet contained the trade volume in Euros between a reporter and a partner country. From this data, an adjacency matrix was created, with each dimension representing a country and each element representing the trade volume between countries. The graph showing the exports between countries is shown in Figure 4.3. The Pagerank of each country is shown in Table 4.4.

Table 4.3: Top 10 Exporters in the EU

Total exports (€)	Country
7.89×10^{11}	DE
4.21×10^{11}	FR
2.81×10^{11}	IT
2.75×10^{11}	NL
2.52×10^{11}	BE
2.07×10^{11}	PL
1.97×10^{11}	ES
1.42×10^{11}	AT
1.27×10^{11}	CZ
1.08×10^{11}	SE

Table 4.4: Pagerank of EU exporters

Pagerank	Country
0.1984	DE
0.11925	NL
0.081275	BE
0.07387	FR
0.071992	IT
0.059386	PL
0.050299	ES
0.042771	CZ
0.034515	AT
0.028537	HU

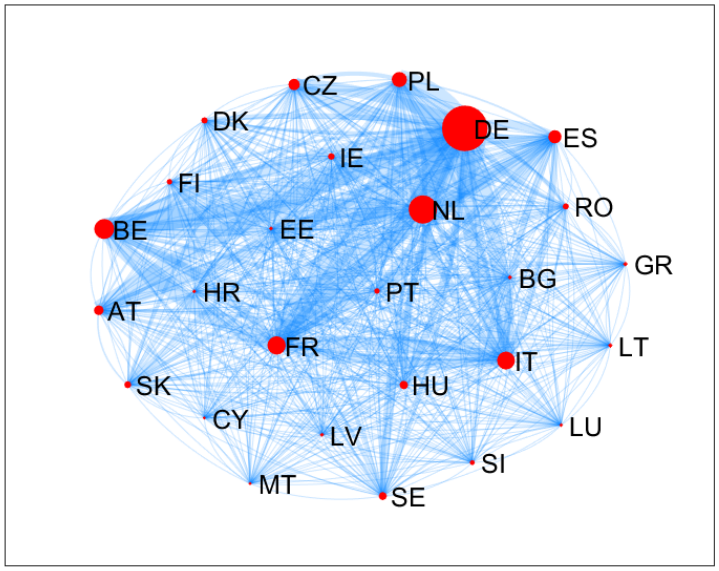

Figure 4.3: Graph showing the flow of net exports between EU members. The nodes represent each country and the edges reflect trade between countries. The size of each node reflects the associated Pagerank. The node labels show the country code. [7]

4.3 Conclusion

In conclusion, the Pagerank algorithm is a powerful tool that can be used to rank web pages based on their importance and relevance. We built the algorithm from its mathematical foundation and then evaluated the reliability and performance

of the algorithm. By applying the algorithm to two different scenarios we have demonstrated its practical application in real-world situations. The Pagerank algorithm can be adapted to different types of networks, making it a versatile tool for various fields.

Appendix A

MATLAB Code

```
function [p] = pagerank(A, alpha, v, iterations)
% PAGERANK Compute the PageRank vector of a directed graph represented by an
% adjacency matrix A. This function has been optimised and makes use of the
% sparsity of A.
%
% Syntax:
%   [p] = pagerank(A, alpha, v, iterations)
%
% Input arguments:
%   - A: An n-by-n sparse adjacency matrix representing the directed graph,
%       where A(i,j) is 1 if there is an edge from node i to node j, and 0
%       otherwise.
%   - alpha: A scalar between 0 and 1 representing the probability that the
%       random surfer will follow a link, rather than jumping to a random
    page.
%   - v: A column vector of length n representing the personalization
%       vector.
%   - iterations: An integer that specifies the number of the power method
%       iterations.
%
% Output arguments:
%   - p: Pagerank vector which ranks the nodes in a graph based on its
%       importance.
n = size(A, 1);
e = ones(n, 1);
column_sum = e' * A;
zero_columns = find(column_sum==0);
zero_col_num = length(zero_columns);
a = sparse(zero_columns, ones(zero_col_num, 1), ones(zero_col_num, 1), n, 1);
column_sum(column_sum == 0) = 1;
p = e/n;
for k = 1:iterations
    p = alpha*(A./column_sum)*p + v*(alpha*a'*p - alpha + 1);
end
end
```

Listing A.1: pagerank.m

```
function [adjacencyMatrix, linkNames] = webCrawler(websiteName, maxDepth,
    maxNodes, blacklist, whitelist)
% webCrawler - crawl a website and generate an adjacency matrix of links
%
% Syntax:
% [adjacencyMatrix, linkNames] = webCrawler(websiteName, maxDepth, maxNodes,
    blacklist, whitelist)
%
% Inputs:
% websiteName - the starting website URL
% maxDepth - maximum depth of the crawl (default 2)
% maxNodes - maximum number of nodes to visit (default 100)
% blacklist - list of URLs to ignore
```

```
% whitelist - list of URLs to only include
%
% Outputs:
% adjacencyMatrix - adjacency matrix of links between visited websites
% linkNames - cell array of URLs of visited websites
    if nargin < 2
        maxDepth = 2;
    end
    if nargin < 3
        maxNodes = 100;
    end
    if nargin < 4
        blacklist = string([]);
    end
    if nargin < 5
        whitelist = string([]);
    end
    visitedWebsites = containers.Map;% Matlabs version of containers
    adjacencyList = containers.Map;
    currentUrl = websiteName;
    errors = string([]);
    visitQueue = {struct('url', websiteName, 'depth', 0)};
    progressBarWidth = 50;
    previousStringLength = progressBarWidth;
    index = 0;
    while ~isempty(visitQueue) && length(visitedWebsites) < maxNodes
        index = index + 1;
        current = visitQueue{1};
        visitQueue(1) = [];
        currentUrl = current.url;
        currentDepth = current.depth;
        if visitedWebsites.isKey(currentUrl) || any(contains(currentUrl,
             blacklist)) || ...
           (~isempty(whitelist) && ~any(contains(currentUrl, whitelist))) &&
               index ~= 1
            continue;
        end
        try
            htmlContent = webread(currentUrl);
            visitedWebsites(currentUrl) = true;
            links = extractLinks(htmlContent);
            for i = 1:length(links)
                link = links{i};
                if startsWith(link, "/")
                    strippedUrl = erase(currentUrl, "https://");
                    if contains(strippedUrl, "/")
                        strippedUrl = extractBefore(strippedUrl, "/");
                    end
                    link = "https://" + strippedUrl + link;
                end
                if ~visitedWebsites.isKey(link) && ...
                   ~any(contains(link, blacklist)) && ...
                   (isempty(whitelist) || any(contains(link, whitelist)))
                    if ~adjacencyList.isKey(currentUrl)
                        adjacencyList(currentUrl) = string([]);
                    end
                    adjacencyList(currentUrl) = [adjacencyList(currentUrl),
                         link];
                    if currentDepth < maxDepth
                        visitQueue{end+1} = struct('url', link, 'depth',
                             currentDepth + 1);
                    end
                end
            end
        catch
            errors = [errors, currentUrl];
        end

        % Update progress bar
        progress = length(visitedWebsites) / maxNodes;
        filledBlocks = floor(progress * progressBarWidth);
        fprintf(repmat('\b', 1, previousStringLength));
        emptyBlocks = progressBarWidth - filledBlocks;
        info = "[" + repmat('=', 1, filledBlocks) + repmat(' ', 1,
```

```
            emptyBlocks) + "]" +...
            "\nPages visited: " + index + ...
            "\nDepth: " + currentDepth + ...
            "\nAdded websites: " + numel(keys(adjacencyList)) + "/" +
                maxNodes + ...
            "\n" + currentUrl + ...
            "\nErrors:" + numel(errors);
        fprintf(info);
        previousStringLength = strlength(info);
    end
    fprintf("\n\nErrors: %s", errors);
    % Convert the adjacency list to an adjacency matrix
    linkNames = keys(adjacencyList);
    adjacencyMatrix = false(length(linkNames));
    for i = 1:length(linkNames)
        for j = 1:length(linkNames)
            if any(strcmp( adjacencyList(linkNames{i}), linkNames{j} ) )
                adjacencyMatrix(i, j) = true;
            end
        end
    end
end

function links = extractLinks(htmlContent)
    links = {};
    expression = '<a\s+(?:[^>]*?\s+)?href="([^"]*)"';
    matches = regexp(htmlContent, expression, 'tokens');
    for i = 1:length(matches)
        links{end+1} = matches{i}{1};
    end
end
```

Listing A.2: webCrawler.m

Bibliography

[1] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1):107–117, 1998. Proceedings of the Seventh International World Wide Web Conference.

[2] F. Tisseur. Math36001 perron–frobenius theory, 2022.

[3] Taher H. Haveliwala and Sepandar D. Kamvar. The second eigenvalue of the google matrix. In *Stanford University Technical Report*, 2003.

[4] A.N. Langville, A.N. Langville, and C.D. Meyer. *Google's PageRank and Beyond: The Science of Search Engine Rankings*. Princeton University Press, 2006.

[5] David F. Gleich. Pagerank beyond the web. *SIAM Review*, 57(3):321–363, 2015.

[6] Aebel Shajan. Pagerank implementation source code. `https://github.com/Aebel-Shajan/pagerank-application`, Apr 2023.

[7] Eurostat. Eurostat imports dataset. `https://ec.europa.eu/eurostat/databrowser/view/DS-018995__custom_5664444/default/table?lang=en`, 2023. Accessed: 2023-04-15.