The University of Manchester

MATH30000

PageRank
The Mathematics Behind Google

Author
Jess Buck 10149726

Supervisor
Françoise Tisseur

May 11, 2020

Contents

---

1 Background

What happens when you type something into Google?

- Depending on the words in your query, a list of relevant web pages is produced.
- The list is sorted into the order believed to be most useful for the user.

In a search engine, the first step is carried out by the query module, and then the pages are passed to the ranking module to be ordered. Here, each page receives a **content score** (how relevant the page is to the query) and a **popularity score** (independent of the query), which are combined to give an **overall score**, by which the pages will be ordered.

With an estimated 5.5 billion Google searches every day [1], and 58 billion web pages in Google's index alone [2], the problem of sorting these pages into a way that's of use to the web surfer remains a key aspect in the design of search engines. As the number of web pages available online continues to increase by many orders of magnitude, whilst the likelihood that a user looks beyond the first 10-20 search results stays constant, clearly the development and implementation of a highly precise sorting algorithm remains to be of great importance [3]. Behind the huge success of Google lies PageRank, an algorithm for computing the popularity score of a page, which was developed in 1998 by Google founders Larry Page and Sergey Brin. Like many of the algorithms used by other popular search engines, PageRank uses link analysis to determine which of the relevant pages will be of most interest to the user. Roughly speaking, *PageRank's thesis is that a webpage is important if it is pointed to by other important pages* [4]. In other words, a page receives a high rank if the sum of its back-links is high, covering both cases that there are a few important pages linking to it, or that there are many pages of lesser importance linking to it [5].

We will see how this relates mathematically to topics in linear algebra, graph theory and Markov chains, and also how the problem can be made formal through a surprisingly simple formula. In addition to this, I will look at methods for solving the problem, ways in which the computation time may be reduced, and the effects of varying some parameters in the model. A lot of research has gone into the science of 'Who Links to Whom' [4], and it turns out much of the theory behind PageRank can be applied to a whole range of domains, including social networks, the ranking of sports players, science, engineered systems, literature, and analysis of road networks. I will investigate the use of PageRank in ranking players performance in sport, by constructing a model for rating football players in a particular football match. I will then be able to compare my results with other ratings from this game that were published online, to see if a simple PageRank model can provide a reasonably accurate ranking when viewed alongside these other more complex rating systems. Finally I will briefly discuss a couple of other applications,

namely in road networks and ecology, to see how these relatively straightforward ideas in mathematics can apply to, and provide insight, in a broad range of topics.

2 The Basic Model

2.1 Creating The System

The first step in setting this out as a mathematical problem is to imagine The Web as a directed graph, in which the nodes represent web pages and each directed edge represents a hyperlink that exists from one web page to another. See Figure 1 below for a very simplified model of The Web, consisting of just 4 pages, labelled 1, 2, 3 & 4. In this section of The Web, for example, page 1 contains hyperlinks to pages 2, 3, & 4, and is linked to by page 4.

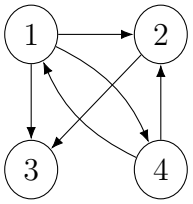

Figure 1: A directed graph of web pages

Using the underlying assumption that the more important a website is, the more links it has from other important websites, the first thing Brin and Page did to determine the PageRank of a page P_i, denoted $r(P_i)$, was to take the sum of the PageRank's of all pages pointing into P_i. An obvious problem with this idea is that these values are unknowns, and so instead, they had to take an iterative approach. The starting point for the iterative process came from the assumption that all web pages had equal PageRank, say $\frac{1}{n}$, where n is the total number of pages in Google's index. This gave the following iterative procedure:

$$r_{k+1}(P_i) = \sum_{P_j \epsilon W_i} \frac{r_k(P_j)}{|P_j|}. \tag{1}$$

Here, W_i is the set of pages pointing into P_i and $|P_j|$ is the number of links leaving page P_j. Here, we know that $|P_j| \neq 0$ because the sum is taken over $P_j \epsilon W_i$, which by definition must have at least 1 outlink (as we know they link to P_i). This iterative process was initiated with $r_0(P_i) = \frac{1}{n}$ for all pages P_i, and the hope was that it would converge to some values, the PageRank for each page. Of course calculating the rank of each page one by one is not practical for real-world applications, and so in order to determine these values for all indexed pages at the same time, the summation was replaced with a matrix. This new system then consists of H (an $n \times n$ matrix), π^T (a $1 \times n$ row vector), and at each iteration we approximate the PageRank vector ($1 \times n$).

H is a row normalized hyperlink matrix with $H_{ij} = \frac{1}{|P_i|}$ if there is a link from node i to j, and 0 otherwise. The non-zero elements of row i correspond to the outlinking pages of page i, whilst the non-zero elements of column i correspond to the inlinking pages of page i. Then $\pi^{(k)T}$ is the PageRank vector at the k^{th} iteration:

$$\pi^{(k+1)T} = \pi^{(k)T} H. \tag{2}$$

Using the example graph in Figure 1, we obtain the matrix H:

$$\begin{pmatrix} 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 \end{pmatrix}$$

We can see here, for example, that as node 1 in Figure 1 has 3 outlinks, to nodes $2, 3, \&4$, that each H_{1j} for $j = 2, 3, 4$ takes a value of $\frac{1}{3}$. Next, applying the iterative formula (2) with starting values $\frac{1}{n} = \frac{1}{4}$, gives the PageRank vector after the first iteration:

$$\begin{pmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{pmatrix} \begin{pmatrix} 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{8} & \frac{5}{24} & \frac{1}{3} & \frac{1}{12} \end{pmatrix}. \tag{3}$$

As with any iterative process, the hope was that for each webpage, this would converge to a unique PageRank vector, one that made sense in the context, and ideally with a relatively low number of computations. However there were some issues with this first basic model. In particular, for the example above, consider what happens to the PageRank vector after carrying out just 5 iterations:

Table 1: First 5 iterates of the PageRank vector

k	$\pi^{(k)T}$
0	$\begin{pmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{pmatrix}$
1	$\begin{pmatrix} \frac{1}{8} & \frac{5}{24} & \frac{1}{3} & \frac{1}{12} \end{pmatrix}$
2	$\begin{pmatrix} \frac{1}{24} & \frac{1}{12} & \frac{1}{4} & \frac{1}{24} \end{pmatrix}$
3	$\begin{pmatrix} \frac{1}{48} & \frac{5}{144} & \frac{7}{72} & \frac{1}{72} \end{pmatrix}$
4	$\begin{pmatrix} \frac{1}{144} & \frac{1}{72} & \frac{1}{24} & \frac{1}{144} \end{pmatrix}$
5	$\begin{pmatrix} \frac{1}{288} & \frac{5}{864} & \frac{7}{432} & \frac{1}{432} \end{pmatrix}$

From Table 1 we can see that each value of $\pi^{(k)T}$ is tending towards 0 with each iteration - instead of a meaningful Pagerank value. This is largely due to the the fact that

H is very **sparse** (many of the elements are zeros). More generally, this process would create **rank sinks**, pages that accumulated more and more PageRank at each iteration and meant that some nodes would get no share. Clearly this is a problem because two pages both with a ranking of 0 are impossible to compare, therefore ideally we'd like the PageRank vector to contain only positive entries. There was also the issue of **cycles**. To see this, consider the very simple case below:

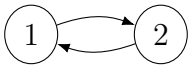

Figure 2: A simple cycle

For a starting vector of $\pi^{(0)T} = (1, 0)$, the iterates $\pi^{(k)T}$ oscillate between $(1, 0)$ and $(0, 1)$ and refuse to converge no matter how many iterations we carry out. To overcome these problems, various changes had to be made to the model.

2.2 Link to Markov Chains

The matrix H contains zero rows for dangling nodes - the web pages with no outlinks, and all other rows are **stochastic** - meaning the entries of each row sum to 1. Recall that a **Markov matrix**, or a (row) stochastic matrix, is a square matrix whose rows are probability vectors - vectors with non-negative entries which sum to 1. So one can observe that H looks a lot like a stochastic transition probability matrix for a Markov chain. A **Markov chain** is a stochastic process that satisfies the 'Markov property':

$$P(X_{t+1} = S_j | X_t = S_{i_t}, X_{t-1} = S_{i_{t-1}}, ..., X_0 = S_{i_0}) = P(X_{t+1} = S_j | X_t = S_{i_t}) \quad (4)$$

for each $t = 0, 1, 2, ...$ [4]. The basic idea is that the next stage of the process depends only on the current state and not on past events. This memoryless property relates to our problem by a fairly reasonable assumption that in our model, the next web page link chosen does not depend on the previous choices made by the user. We call this chain a **random walk**. Since we know a lot of the theory behind Markov chains, this can be applied to the PageRank problem. For example, we have the following theorem:

Theorem 2.2.1 (Perron, 1907) *If $A \in \mathbb{R}^{n \times n}$ then:*

1. *$\rho(A) > 0$ (positive spectral radius).*
2. *$\rho(A)$ is an eigenvalue of A.*
3. *There is an eigenvector x with $x > 0$ and $Ax = \rho(A)x$.*
4. *The eigenvalue $\rho(A)$ has algebraic multiplicity 1 ($\rho(A)$ is a simple eigenvalue).*

5. *All the other eigenvalues are less than $\rho(A)$ in absolute value, i.e., $\rho(A)$ is the only eigenvalue of maximum modulus.*

Perron's theorem essentially says that a real square matrix with positive elements has a unique largest real eigenvalue, and an associated eigenvector can be chosen to have positive components. If we assume our matrix to be positive, this represents the situation in which every node (representing a web page in our case) is connected to every other node. Moreover, for a row stochastic matrix, we can also apply the following theorem:

Theorem 2.2.2 *Let A be an $n \times n$ row stochastic matrix. Then the following are true:*

- *The stochastic matrix A has an eigenvalue equal to 1.*
- *The absolute value of any eigenvalue of the stochastic matrix A is less than or equal to 1 [4].*

Proof. We can compute that:

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} = \begin{pmatrix} a_{11} + a_{12} + \cdots + a_{1n} \\ a_{21} + a_{22} + \cdots + a_{2n} \\ \vdots \\ a_{n1} + a_{n2} + \cdots + a_{nn} \end{pmatrix} = 1 \cdot \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}, \tag{5}$$

where the last inequality comes from the fact that A is stochastic, and so all rows sum to 1. This shows that 1 is indeed an eigenvalue of A, corresponding to the eigenvector e. We now show that this is the largest eigenvalue of A. So let λ be an eigenvalue of A with corresponding eigenvector v. In other words,

$$Av = \lambda v. \tag{6}$$

Then considering the i-th row of each side, we obtain

$$a_{i1}v_1 + a_{i2}v_2 + \cdots + a_{in}v_n = \lambda v_i, \tag{7}$$

for $i = 1, ..., n$. Let k be such that

$$|v_k| = \max_i |v_i|$$

and note that $|v_k| > 0$ since an eigenvector is always a nonzero vector. Then using (7) with $i = k$, we obtain

$$\begin{aligned} |\lambda| \cdot |v_k| &= |a_{k1}v_1 + a_{k2}v_2 + \cdots + a_{kn}v_n| \\ &\leq a_{k1}|v_1| + a_{k2}|v_2| + \cdots + a_{kn}|v_n| \\ &\leq a_{k1}|v_k| + a_{k2}|v_k| + \cdots + a_{kn}|v_k| \\ &= (a_{k1} + a_{k2} + \cdots + a_{kn})|v_k| = |v_k|. \end{aligned} \tag{8}$$

In the above, the second line uses the triangle inequality and the fact that $a_{ij} \geq 0$, whilst the third comes from the fact that $|v_k|$ is maximal. It then follows easily that $|\lambda| \leq 1$, as required. □

By this theorem, if we assume that A is stochastic, then we know that 1 is an eigenvalue and all the other eigenvalues are in modulus strictly less than 1. Therefore using Perron's theorem and what we know about Markov thoery, we can deduce that for any starting vector, the power method (see section 4.1) applied to a positive Markov matrix M will converge to a unique positive vector called the stationary vector. Hence by modifying H to be a Markov matrix with these properties, we can hope to obtain a convergent iteration.

2.3 The Random Surfer

In their early papers, Brin and Page used the notion of a random surfer instead of viewing the PageRank problem in the context of Markov Chains. The idea behind this is to imagine an internet surfer following hyperlinks through the web at random, and continuing to do so indefinitely. In the long run, the proportion of time the random surfer spends on a given page is a measure of the relative importance of that page. [4] However when the surfer finds themselves on a page with no outlinks (dangling nodes), clearly their journey comes to an end. Alternatively, it is a reasonable assumption to make that a web surfer moving through pages by following hyperlinks will at some point lose interest and instead access a new page entirely. In this context, this issue is the reason for making a stochasticity adjustment to the model.

3 Changes to The Model

3.1 The Stochasticity Adjustment

To overcome the problem of the iteration getting stuck on dangling nodes, a common feature of the real web, we must make the matrix H fully stochastic. To do this we replace zero rows with $\frac{1}{n}e^T$, where e^T is the row vector of all 1s. In the scenario of the random surfer, this represents the idea that if the surfer enters a dangling node, they then jump to any other page, each with equal probability. For the example directed graph in Figure 1 used above, we obtain the stochastic matrix S:

$$\begin{pmatrix} 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & 1 & 0 \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 \end{pmatrix} \tag{9}$$

Now that S is stochastic, it is the transition probability matrix for a Markov Chain.

3.2 The Primitivity Adjustment

In the random surfer argument, the primitivity adjustment represents the surfer stopping following hyperlinks, teleporting at random to a completely new page, and then continuing to follow hyperlinks. To represent this mathematically Brin & Page added the teleportation matrix E, given by $1/nee^T$, by which there is clearly equal probability of teleporting to each of the n pages. This then allows us to construct the new matrix G, a **convex combination** (ie. the coefficients of the two matrices: α and $(1-\alpha)$, sum to 1) of the two stochastic matrices S and E. This is known as the **Google matrix**:

$$G = \alpha S + (1-\alpha)\frac{1}{n}ee^T, \tag{10}$$

where α is a scalar between 0 and 1, a parameter that controls the amount of time the surfer follows the hyperlinks instead of teleporting. So at each choice, the surfer chooses an outlink at random with probability α and chooses a random page anywhere on the web with probability $(1-\alpha)$. This constant α is often called the **damping factor** (see section 5.1).

Note that using this choice for the personalisation vector (see section 5.3) ensures that the Google matrix is positive - which implies primitivity. Now G is a stochastic and positive matrix, it has the required properties to guarantee a convergent stationary vector for any starting point. Put simply, Google's PageRank method is just the power method applied to G:

$$\pi^{(k+1)T} = \pi^{(k)T}G. \tag{11}$$

3.3 Simple example

Continuing with the example of the small web graph used throughout, and taking α to be 0.85, we obtain the google matrix G:

$$\begin{aligned} G &= 0.85 \times S + 0.15 \times \frac{1}{4} \times ee^T \\ &= 0.85 \times \begin{pmatrix} 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & 1 & 0 \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 \end{pmatrix} + \frac{3}{80} \times \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix} \\ &= \begin{pmatrix} \frac{3}{80} & \frac{77}{240} & \frac{77}{240} & \frac{77}{240} \\ \frac{3}{80} & \frac{3}{80} & \frac{71}{80} & \frac{3}{80} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{37}{80} & \frac{37}{80} & \frac{3}{80} & \frac{3}{80} \end{pmatrix}. \end{aligned} \tag{12}$$

Then using equation (11) above, we can experiment performing some iterations of the method, using a starting vector of $\pi^T = (\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4})$. The values obtained from the first 7 iterations are displayed below:

Iteration	π_1	π_2	π_3	π_4
0	0.25	0.25	0.25	0.25
1	0.1969	0.2677	0.3740	0.1615
2	0.1856	0.2414	0.4003	0.1727
3	0.1960	0.2486	0.3803	0.1751
4	0.1926	0.2483	0.3851	0.1738
5	0.1932	0.2478	0.3850	0.1740
6	0.1932	0.2480	0.3847	0.1741
7	0.1932	0.2480	0.3848	0.1740

In addition, we can obtain the stationary vector of G, and find it is given by:

$$\pi^T = \begin{pmatrix} 0.1932 & 0.2480 & 0.3848 & 0.1740 \end{pmatrix}. \tag{13}$$

We observe that after just 7 iterations, the values obtained for π^T agree with the stationary vector of G to 4 decimal places, but in real-world applications involving the web and its billions of pages, a much higher number of iterations would be needed to reach the degree of accuracy required.

So π^T given above is the PageRank vector of the simple model web defined in Figure 1. We recall that the output of the PageRank algorithm is a probability distribution, representing the likelihood of ending up on each of the pages if a surfer goes through the web randomly following links. Therefore the significance of this is that when randomly surfing this web, 19.32% of your time will be spend on Page 1, 24.8% of your time on Page 2, etc... Hence we have determined that using the PageRank model, Page 3 appears to be the most important page, followed by Page 2, and Page 4 has the least importance.

Note that this example involves just 4 webpages, as opposed to Google's index with an estimated 58 billion (as of May 2020). The reason these computations are even possible is that the matrix G depends on the matrix H which is very sparse, as most pages will only link to a few other pages in the web. Estimates show that the average webpage has about 10 outlinks [4], and so H will have approximately $10n$ non zero elements, which, for large n, is significantly smaller than the n^2 nonzero elements of a completely dense matrix.

4 Solving the System

In terms of computing the solution, our problem can be stated in 2 ways:

1. Solve the eigenvector problem for π^T:

$$\pi^T = \pi^T G. \tag{14}$$

Here the objective is to find the normalised dominant left-hand eigenvector of G, corresponding to the dominant eigenvalue. We know that the dominant eigenvalue of a stochastic matrix (such as G) will be equal to 1 (see Theorem 2.2.2).

2. Solve the linear homogeneous system for π^T:

$$\pi^T(I - G) = 0^T. \tag{15}$$

In this system we are finding the left-hand null vector of $(I - G)$.

Both systems are also subject to the condition $\pi^T e = 1$, which normalises π and so ensures it is a probability vector. The ith element of π^T, π_i, is the PageRank of page i. A lot of research has been done on the topic of Markov Chains, and there are many different methods for finding the stationary vector π^T.

4.1 The Power Method

One method found to be particularly advantageous in our problem, due to the nature of the Google matrix G, is the Power Method. The power method is a technique for finding an eigenvector of a square matrix associated with the largest eigenvalue in modulus, and is one of the simplest and oldest methods for doing so [4]. Despite being somewhat slow in comparison with other iterative methods, it is very easy to implement and program. What's more, it is storage friendly, and is what's known as *matrix-free*, meaning that neither of the matrices S and G must be formed or stored. In fact, we find that it is possible to express equation (11) in terms of the very sparse matrix H, and this turns out to be vital in limiting the number of computations and storage required in evaluating the solution. This can be done as follows:

$$\begin{aligned} \pi^{(k+1)T} &= \pi^{(k)T} G \\ &= \alpha \pi^{(k)T} S + \frac{1-\alpha}{n} \pi^{(k)T} e e^T \\ &= \alpha \pi^{(k)T} H + (\alpha \pi^{(k)T} a + 1 - \alpha) e^T / n. \end{aligned} \tag{16}$$

Written in this form, one can see that the vector-matrix multiplications ($\pi^{(k)T} H$) are executed on the matrix H, which is very sparse due to the nature of the web, and the matrices S and G don't need to be formed or stored. This is very useful as modifying and storing elements of G is much more complex and requires greater storage.

One final benefit of this method primarily using the matrix H, is that due to its sparsity, the number of iterations required to converge to a satisfactory approximation is low, roughly 50-100 matrix-vector multiplications [4]. It turns out the speedy convergence of the method is related to the structure of G. This is due to the fact that the **asymptotic rate of convergence** of the power method applied to a matrix depends on the ratio of the two eigenvalues largest in size. Indeed, it is the rate at which $|\lambda_2/\lambda_1|^k \to 0$. To see why this is the case, we consider the following piece of Markov theory [6]:

Theorem 4.1.1 (Convergence of the Power Method) *If A is an $n \times n$ diagonalisable matrix, then there exists a nonzero vector x_0 such that the sequence of vectors given by*

$$A_0, A^2x_0, A^3x_0, \ldots, A^kx_0, \ldots,$$

approaches a multiple of the dominant eigenvector of A.

Proof. Using the fact that a diagonalisable $n \times n$ matrix has n linearly independent eigenvectors $x_1, \ldots, x_n$ (which form a basis for $\mathbb{R}^n$), with respective eigenvalues $\lambda_1, \ldots, \lambda_n$, we order these eigenvalues such that λ_1 is the dominant eigenvalue. For the initial approximation x_0, we choose a nonzero vector such that the linear combination

$$x_0 = c_1x_1 + c_2x_2 + \cdots + c_nx_n$$

has nonzero leading coefficients, and multiply both sides of the equation by A to give:

$$\begin{aligned} Ax_0 &= A(c_1x_1 + c_2x_2 + \cdots + c_nx_n) \\ &= c_1(Ax_1) + c_2(Ax_2) + \cdots + c_n(Ax_n) \\ &= c_1(\lambda_1x_1) + c_2(\lambda_2x_2) + \cdots + c_n(\lambda_nx_n). \end{aligned} \tag{17}$$

Note that when choosing an initial approximation, we cannot have $c_1 = 0$, as in this case the power method may not converge. Next, repeatedly multiplying both sides by A we obtain

$$\begin{aligned} A^kx_0 &= c_1(\lambda_1^kx_1) + c_2(\lambda_2^kx_2) + \cdots + c_n(\lambda_n^kx_n) \\ &= \lambda_1^k\left(c_1x_1 + c_2\left(\frac{\lambda_2}{\lambda_1}\right)^k x_2 + \ldots c_n\left(\frac{\lambda_n}{\lambda_1}\right)^k x_n\right). \end{aligned} \tag{18}$$

Using our assumption that λ_1 is the eigenvalue largest in absolute value, it follows that

$$\left|\frac{\lambda_i}{\lambda_1}\right| < 1$$

for $i = 2, 3, \ldots, n$. Therefore, each of these fractions raised to the power of k must tend towards 0 as $k \to \infty$. It follows that the approximation

$$A^kx_0 \approx \lambda_1^kc_1x_1, c_1 \neq 0,$$

improves as k increases. As x_1 is a dominant eigenvector, so is any scalar multiple of x_1. Therefore we have shown that A^kx_0 approaches a multiple of the dominant eigenvector of A. □

The proof above gives us some insight into the rate of convergence of the power method. That is, if the eigenvalues of A are ordered such that

$$|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_n|,$$

then the power method will converge quickly if

$$\left|\frac{\lambda_2}{\lambda_1}\right|$$

is small. Now since we know that $\lambda_1 = 1$ for stochastic matrices such as G, it is therefore $|\lambda_2|$ that controls the rate of convergence. Since G is also positive, $|\lambda_2| < 1$. It can be shown that if the respective spectrums are $\sigma(S) = \{1, \mu_2, ..., \mu_n\}$ and $\sigma(G) = \{1, \lambda_2, ..., \lambda_n\}$, then

$$\lambda_k = \alpha\mu_k, \tag{19}$$

for $k = 2, 3, ..., n$ (See Theorem 4.1.2).

What's more, the link structure of the web makes it likely that $|\mu_2| \approx 1$, and so $|\lambda_2(G)| \approx \alpha$. As a consequence, the damping factor α explains the quick convergence of this method. Taking the value of α still believed to be used by Google of 0.85, we can compute $\alpha^{50} = 0.85^{50} \approx 0.00296$, implying that after 50 iterations we can expect roughly 2-3 places of accuracy in the approximate PageRank vector. In order to distinguish between many similar PageRank scores, a higher lever of accuracy may be required, but once these scores are combined with content scores to give the overall ranking, this often becomes less important.

In the next section, we extend our definition of the Google matrix from $G = \alpha S + (1-\alpha)\frac{1}{n}ee^T$ to $G = \alpha S + (1-\alpha)ev^T$, where $v^T > 0$ is a probability vector known as the **personalisation vector**, to create a more general Google matrix. First, I present a theorem and proof regarding the second eigenvalue of this more general matrix, as referred to in (19) above.

Theorem 4.1.2 *If the spectrum of the stochastic matrix S is $\{1, \lambda_1, \lambda_2, ..., \lambda_n\}$, then the spectrum of the Google matrix $G = \alpha S + (1-\alpha)ev^T$ is $\{1, \alpha\lambda_1, \alpha\lambda_2, ..., \alpha\lambda_n\}$, where v^T is a probability distribution vector, and $\alpha \in (0, 1)$.*

Proof. Since S is stochastic, $(1, e)$ is an eigenpair of S. Let $Q = (e, X)$ be a non-singular matrix that has the eigenvector e as its first column. Let $Q^{-1} = \begin{pmatrix} y^T \\ Y^T \end{pmatrix}$. Then $Q^{-1}Q = \begin{pmatrix} y^Te & y^TX \\ Y^Te & Y^TX \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & I \end{pmatrix}$, which gives two useful identities, $y^Te = 1$ and $Y^Te = 0$. As a result, the similarity transformation

$$Q^{-1}SQ = \begin{pmatrix} y^Te & y^TSX \\ Y^Te & Y^TSX \end{pmatrix} = \begin{pmatrix} 1 & y^TSX \\ 0 & Y^TSX \end{pmatrix} \tag{20}$$

reveals that Y^TSX contains the remaining eigenvalues of S, $\lambda_2, ..., \lambda_n$. Applying the similarity transformation to $G = \alpha S + (1-\alpha)ev^T$ gives

$$\begin{aligned} Q^{-1}(\alpha S + (1-\alpha)ev^T)Q &= \alpha Q^{-1}SQ + (1-\alpha)Q^{-1}ev^TQ \\ &= \begin{pmatrix} \alpha & \alpha y^TSX \\ 0 & \alpha Y^TSX \end{pmatrix} + (1-\alpha)\begin{pmatrix} y^Te \\ Y^Te \end{pmatrix}\begin{pmatrix} v^Te & v^TX \end{pmatrix} \\ &= \begin{pmatrix} \alpha & \alpha y^TSX \\ 0 & \alpha Y^TSX \end{pmatrix} + \begin{pmatrix} (1-\alpha) & (1-\alpha)v^TX \\ 0 & 0 \end{pmatrix} \\ &= \begin{pmatrix} 1 & \alpha y^TSX + (1-\alpha)v^TX \\ 0 & \alpha Y^TSX \end{pmatrix}. \end{aligned} \tag{21}$$

Therefore, the eigenvalues of $G = \alpha S + (1-\alpha)ev^T$ are $\{1, \alpha\lambda_2, \alpha\lambda_3, ..., \lambda_n\}$. □

5 Parameters in the Model

5.1 The Damping Factor

As discussed above, α is a parameter that controls the proportion of time our random surfer spends following hyperlinks as opposed to teleporting to a new page. In the papers of Brin and Page, a value of α of 0.85 is typically taken, and this is the value still assumed to be today. Both increasing and decreasing this value has different effects on the system, and 0.85 is taken as a kind of compromise. Firstly, testing the effects of varying α demonstrates that a lower value, for example 0.7, causes the power method to converge more quickly, whilst a higher value, for example 0.9, causes a slower convergence. One can see this through how the rate of convergence is calculated, as discussed in section 4.1. But another factor to consider is that a higher value reflects a more realistic model of a web surfer, one in which more time is spent following the hyperlink structure of the web, as opposed to randomly jumping to new web pages. Although, it has also been found that as $\alpha \to 1$, the PageRank vectors fluctuate much more for small changes in the structure of the web, i.e. it affects the sensitivity of the resulting vector.
The effect that varying α has on the sensitivity of the solution can be investigated by studying the derivative of π^T with respect to α. Whilst this derivative does not tell us exactly how π^T varies as α does, it provides a good approximation of what will happen. For small values of α, we can easily deduce that the resulting PageRank vector is relatively insensitive to variations in α. Consider the following theorem:

Theorem 5.1.1 *If $\pi^T(\alpha) = (\pi_1(\alpha), \pi_2(\alpha), ..., \pi_n(\alpha))$ is the PageRank vector, then*

$$\left|\frac{d\pi_j(\alpha)}{d\alpha}\right| \leq \frac{1}{1-\alpha} \tag{22}$$

for each j=1,2,...,n, and

$$\left|\left|\frac{d\pi^T(\alpha)}{d\alpha}\right|\right|_1 \leq \frac{2}{1-\alpha}. \tag{23}$$

Proof. We differentiate both sides of $\pi^T(\alpha) = \pi^T(\alpha)(\alpha S + (1-\alpha)ev^T)$ to obtain

$$\begin{aligned}\frac{d\pi^T(\alpha)}{d\alpha} &= \pi^T(\alpha)\frac{d(\alpha S + (1-\alpha)ev^T)}{d\alpha} + \frac{d\pi^T(\alpha)}{d\alpha}(\alpha S + (1-\alpha)ev^T) \\ &= \pi^T(\alpha)(S - ev^T) + \frac{d\pi^T(\alpha)}{d\alpha}(\alpha S) + \frac{d\pi^T(\alpha)}{d\alpha}(1-\alpha)ev^T.\end{aligned} \tag{24}$$

Then recalling the condition on our system of $\pi^T(\alpha)e = 1$, equation (24) simplifes to

$$\frac{d\pi^T(\alpha)}{d\alpha}(I - \alpha S) = \pi^T(\alpha)(S - ev^T).$$

Next note that if the spectral radius of a matrix M is less than 1 then the matrix $I - M$ is invertible (see appendix, Theorem 10.1.1). Therefore matrix $I - \alpha S(\alpha)$ is non-singular because $\alpha < 1$ guarantees that $\rho(\alpha S(\alpha)) < 1$, and so we can post-multiply the above equation by $(I - \alpha S)^{-1}$ to give

$$\frac{d\pi^T(\alpha)}{d\alpha} = \pi^T(\alpha)(S - ev^T)(I - \alpha S)^{-1}. \tag{25}$$

Now we note that for every real $x \in e^{\perp}$ (the orthogonal complement of span$\{e\}$), and for all real vectors $y \in \mathbb{R}^n$,

$$|x^T y| \leq ||x||_1 \left(\frac{y_{\max} - y_{\min}}{2}\right). \tag{26}$$

This is because for all real α, and using the fact that $x^T e = 0$ for $x \in e^{\perp}$,

$$|x^T y| = |x^T(y - \alpha e)| \leq ||x||_1 ||y - \alpha e||_\infty,$$

by Hölder's inequality. Furthermore, $\min_\alpha ||y - \alpha e||_\infty = (y_{max} - y_{min})/2$, where the minimum is attained when $\alpha = (y_{max} + y_{min})/2$. Now considering the j^{th} component in equation (25), it follows that

$$\frac{d\pi_j(\alpha)}{d\alpha} = \pi^T(\alpha)(S - ev^T)(I - \alpha S)^{-1}e_j,$$

where e_j is the j^{th} standard basis vector (i.e, the j^{th} column of $I_{n\times n}$). Since we have that $\pi^T(\alpha)(S - ev^T)e = 0$, we may apply inequality (26), taking $y = (I - \alpha S)^{-1}e_j$ and $x^T = \pi^T(\alpha)(S - ev^T)$, to obtain

$$\left|\frac{d\pi_j(\alpha)}{d\alpha}\right| \leq ||\pi^T(\alpha)(S - ev^T)||_1 \left(\frac{y_{max} - y_{min}}{2}\right). \tag{27}$$

Next we recall that $\pi(\alpha)$ is a probability vector, so $\pi^T(\alpha)e = 1$, or $||\pi^T(\alpha)||_1 = 1$. Therefore,

$$\pi^T(\alpha)(S - ev^T) = \pi^T(\alpha)S - \pi^T(\alpha)ev^T = \pi^T(\alpha)S - v^T,$$

and so

$$||\pi^T(\alpha)(S - ev^T)||_1 = ||\pi^T(\alpha)S - v^T||_1 \leq ||\pi^T(\alpha)S||_1 + ||v^T||_1,$$

by the triangle inequality. But $||\pi^T(\alpha)S||_1 \leq ||\pi^T(\alpha)||_1||S||_1 = 1$ and $||v^T||_1 = 1$, and so we see that $||\pi^T(\alpha)(S - ev^T)||_1 \leq 1 + 1 = 2$, so then by (27),

$$\left|\frac{d\pi_j(\alpha)}{d\alpha}\right| \leq y_{\max} - y_{\min}.$$

Now we use the fact that $(I - \alpha S)^{-1} \geq 0$ (see appendix, Theorem 10.1.2 with $X = \alpha S$), together with the observation that

$$(I - \alpha S)e = (1 - \alpha)e \implies (I - \alpha S)^{-1}e = (1 - \alpha)^{-1}e,$$

to conclude that $y_{\min} \geq 0$ and

$$y_{\max} \leq \max_{i,j}[(I - \alpha S)^{-1}]_{i,j} \leq ||(I - \alpha S)^{-1}||_\infty = ||(I - \alpha S)^{-1}e||_\infty = \frac{1}{1 - \alpha}.$$

Consequently,

$$\left|\frac{d\pi_j(\alpha)}{d\alpha}\right| \leq \frac{1}{1 - \alpha},$$

proving the first part of the theorem. The second result is a direct consequence of (25), along with the above observation that

$$||(I - \alpha S)^{-1}||_\infty = ||(I - \alpha S)^{-1}e||_\infty = \frac{1}{1 - \alpha}.$$

□

What this theorem tells us is that for small values of α, the PageRank values are not very sensitive as a function of α. However for larger values of α, i.e. as $\alpha \to 1$, the upperbound for the derivative, given by $\frac{1}{1-\alpha}$, becomes increasingly meaningless, as it tends towards infinity. As the sensitivity of the PageRank vector when evaluated with larger values of α are of more importance to us than for smaller values, we also consider the following theorem.

Theorem 5.1.2 *If $\pi^T(\alpha)$ is the PageRank vector associated with the Google matrix $G(\alpha) = \alpha S + (1 - \alpha)ev^T$, then*

$$\frac{d\pi^T(\alpha)}{d\alpha} = -v^T(I - S)(I - \alpha S)^{-2}. \tag{28}$$

Proof. We first rearrange $\pi^T(\alpha) = \pi^T(\alpha)(\alpha S + (1 - \alpha)ev^T)$ to obtain

$$0^T = \pi^T(\alpha)(I - \alpha S - (1 - \alpha)ev^T).$$

Then multiply on the right-hand side by $(I - \alpha S)^{-1}$ to give

$$0^T = \pi^T(\alpha)(I - (1-\alpha)ev^T(I - \alpha S)^{-1}) \implies \pi^T(\alpha) = (1-\alpha)v^T(I - \alpha S)^{-1}.$$

Next we note that for an invertible matrix $A(\alpha)$, we have

$$\frac{dA(\alpha)^{-1}}{d\alpha} = -A^{-1}(\alpha)\frac{dA(\alpha)}{d\alpha}A^{-1}(\alpha),$$

(see appendix, Theorem 10.1.3). Using this along with the fact that $(I - S)$ commutes with $(I - \alpha S)^{-1}$ gives

$$\begin{aligned}
\frac{d\pi^T(\alpha)}{d\alpha} &= (1-\alpha)v^T(I - \alpha S)^{-1}S(I - \alpha S)^{-1} - v^T(I - \alpha S)^{-1} \\
&= -v^T(I - \alpha S)^{-1}(I - (1-\alpha)S(I - \alpha S)^{-1}) \\
&= -v^T(I - \alpha S)^{-1}(I - \alpha S - (1-\alpha)S)(I - \alpha S)^{-1} \\
&= -v^T(I - \alpha S)^{-1}(I - S)(I - \alpha S)^{-1} \\
&= -v^T(I - S)(I - \alpha S)^{-2}.
\end{aligned} \tag{29}$$

□

We can also take from this result the limiting values of this derivative, indeed

$$\lim_{\alpha \to 0} \frac{d\pi^T(\alpha)}{d\alpha} = -v^T(I - S),$$

and

$$\lim_{\alpha \to 1} \frac{d\pi^T(\alpha)}{d\alpha} = -v^T(I - S)^{-1}.$$

It is worth noting that the PageRank vector is a rational function in α, and so this derivative indeed exists. Consider Cramer's rule [8] for the solution of the linear system $Ax = b$,

$$x_i = \frac{\det(A_i)}{\det(A)}.$$

where $A_i = A$ with the ith column replaced by b. We see that each component of the solution is a ratio of determinants. It is the case that the determinant is a polynomial in the matrix entries. If each entry in the matrix depends on a parameter, such as α as in $(I - \alpha S)$, then the determinant will be a polynomial in α. Replacing a column with $b = (1-\alpha)v$ does not change this for A_i meaning we can conclude that the PageRank vector is a rational function of α.

In section 6.3 I make use of this theorem to demonstrate what happens to the sensitivity of the PageRank vector as $\alpha \to 1$.

5.2 The Hyperlink Matrix H

First we discuss our earlier approach for filling out the values of the hyperlink matrix H. Recall we distributed equal probability to each outgoing link from a web page, for example for a page with hyperlinks to 4 different pages, we assumed a 1/4 chance of following each of these links. As long as each row represents a probability distribution (ie. the values sum to 1), then H will be stochastic, and so these values do not necessarily need to be chosen in this way. Whilst doing this is a democratic and fairly logical approach, in reality a web surfer may be far more likely to visit certain pages than others. This could be based on factors such as their current location, interests, page-content, advertising etc. One approach that addresses this consideration is to use access logs to study surfer tendencies, using the data from other web user's surfing history to modify our probabilities that certain pages will be visited. In addition to this, many other methods for adjusting the values in in the hyperlink matrix H, have been proposed.

5.3 The Teleportation Matrix E

One of the first major changes to the model suggested by Brin & Page was regarding the teleportation matrix E, which we have so far taken to be equal to $1/n ee^T$. In place of this, they suggested using ev^T, for $v^T > 0$ the **personalisation** or **teleportation vector**. Since v^T is a probability vector with positive entries, our earlier theory still applies and so a stationary vector associated with the Markov chain exists, namely our PageRank vector. The difference is that the probabilities of randomly teleporting to a new page are no longer uniformly distributed, and instead follow the distribution of v^T, whatever we set this up to be. As a very simple example, consider a network consisting of 4 pages, in which page 1 has no outgoing weblinks. Through the simple model, we would take the first row of the Google matrix, denoted here as g_1, to be:

$$g_1^T = \begin{pmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{pmatrix}. \tag{30}$$

However if web usage logs show that a particular class of surfer may be more likely to jump to pages 2 or 3 next, then it may be more suitable to take as g_1 the personalised vector

$$g_1^T = \begin{pmatrix} \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{pmatrix}. \tag{31}$$

So the benefit of using this vector is that each web surfer can have their own personalised probability distribution which is tailored to their habits, for example a sports fan can bias their v^T vector so that v_i is large for sport-filled pages P_i, and v_j is almost 0 for all other pages. Fortunately to produce a PageRank that is personalised for a particular user, only the constant vector v_T added at each iteration must be modified. Similarly, for the linear system formulation of the PageRank problem only the right-hand side of the system changes for various personalized vectors. This is highly beneficial as it means it is relatively straightforward to create personalised PageRanks, and these can be used in a wide range of applications, including personal search engines. Search engines such

as these may only require the simple input of a user's bookmarks, or homepage, and then when using the search engine the user will see results tailored to their interests [5].

6 $H200$ Matrix Example

In order to test some of the theory outlined above, I decided to carry out some experiments using a much larger and more realistic hyperlink matrix, namely the $H200$ matrix. This 200×200 matrix corresponds to a small section of The Web, with root at `http://www.mathworks.com`. This hyperlink matrix was created from the adjacency graph of this section of The Web, using the MATLAB function 'surfer.m'. This function starts at the URL root and follows web links until we obtain an adjacency graph, with n nodes. The function returns H, or in this case $H200$, which is an $n \times n$ hyperlink matrix, with $H_{ij} = 1$ if node j links to node i, and $H_{ij} = 0$ otherwise.

6.1 Sparsity

As is the case for web hyperlink matrices in general, we expect $H200$ to be very sparse due to the nature of The Web. Indeed, I created a plot in MATLAB to display the sparsity pattern of this matrix. I did this using the command 'spy(H200)', which makes use of the MATLAB 'spy' function. This obtained the graph shown in Figure 3. Note that below the plot is printed '$nz = 2978$', where here MATLAB has computed the total number of nonzero entries in the $H200$ matrix, i.e. the number of points plotted in this graph. We observe that this value is just 2978, out of a possible $200 \times 200 = 40000$ entries.

6.2 Computing the PageRank vector

Next, I wanted to carry out the power method on the Google matrix for this portion of the web. The first step was to form the stochastic matrix S corresponding to my hyperlink matrix $H200$. I then used MATLAB to carry out iterations of the power method, until my PageRank vector had converged to a specified tolerance. For this power method function, I used standard values of $\alpha = 0.85$, and v to be the standard personalisation vector consisting of $n = 200$ values, each of $\frac{1}{n}$. See appendix 10.2 for the MATLAB code I used to form the stochastic matrix S, as well as to carry out the power method. I decided to use the bar function of MATLAB to plot the resulting PageRank vector. This is displayed in Figure 4 below.

In the function used for computing the PageRank vector via the power method, 'cnt' is used to count the number of iterations required. In this case I returned a value of '$cnt = 36$', meaning that the power method took 36 iterations for the PageRank vector to converge to a tolerance of 10^{-5}. We have seen in Section 4.1 that the rate of convergence

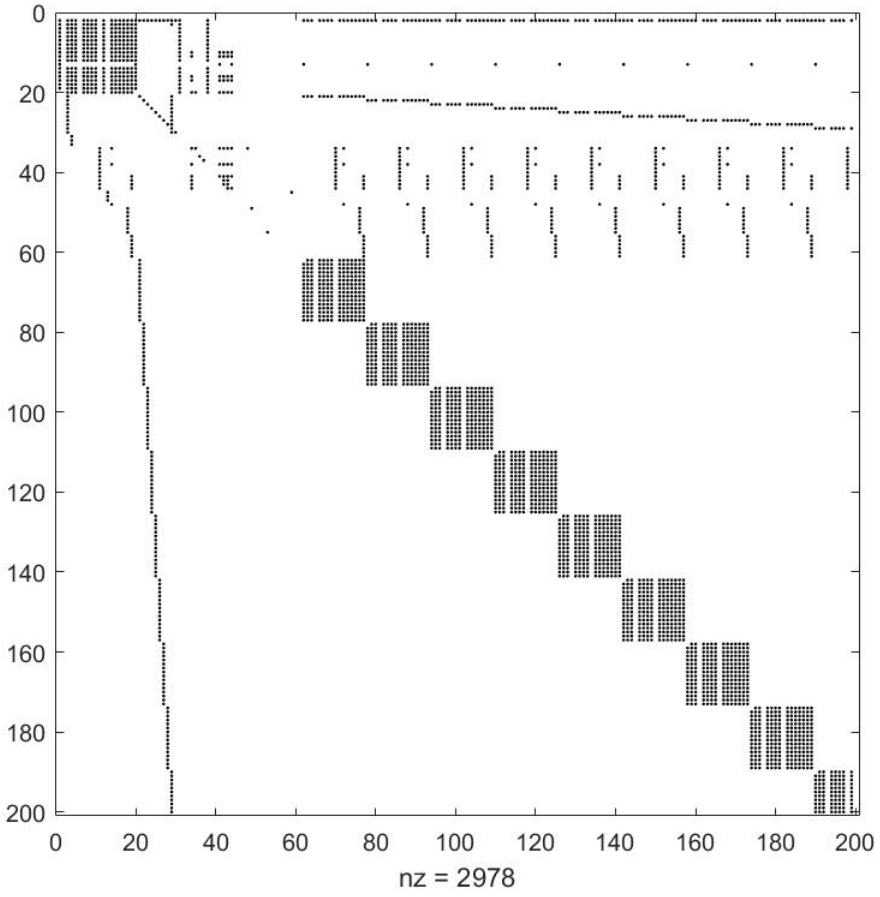

Figure 3: Plot displaying the sparsity pattern of the H200 matrix

of the power method applied to the Google Matrix depends on the absolute value of the second largest eigenvalue, $|\lambda_2(G)|$, and the method will converge quickly is this value is small. We also saw that $|\lambda_2(G)| \approx \alpha$. I decided to verify this using the $H200$ matrix. To do this I computed the Google Matrix of the sparse $H200$ hyperlink matrix, using a value of α of 0.85, and found the eigenvalue second largest in modulus to be 0.8387 (see end of 10.2 in appendix). As clearly $0.8387 \approx 0.85$, this supports the claim. So we know that an estimate for the number of iterations required for convergence is given by

$$\frac{log(\tau)}{\log(|\lambda_2(G)|)},$$

where τ is a specified tolerance level. Therefore in this case, we expect roughly

$$\frac{log(10^{-5})}{\log(0.8387)} = 65.4507...$$

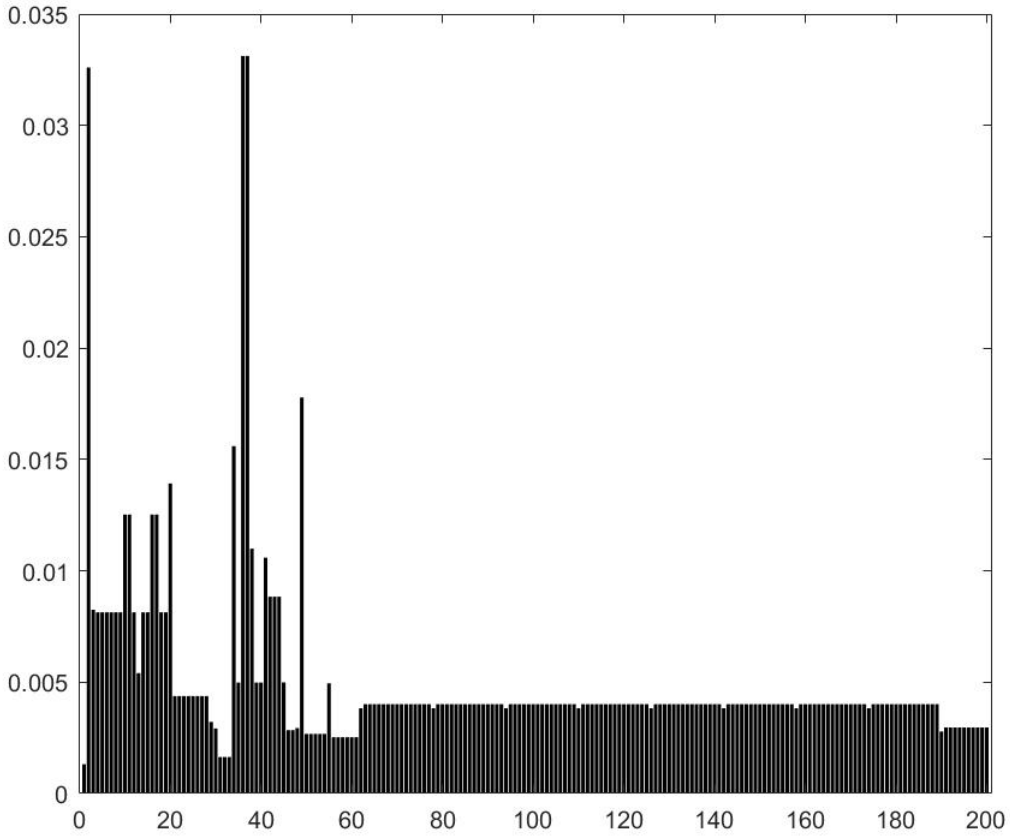

Figure 4: Bar plot of the PageRank vector for the H200 matrix

iterations for the PageRank vector of the $H200$ matrix to convergence to the tolerance of 10^{-5}. Note our value obtained of 36 is somewhat better than this, but for a more realistic web graph containing millions of nodes as opposed to 200, we would most likely expect a slower convergence, closer to 65 iterations.

6.3 Sensitivity

Finally, following on from Theorem 5.1.2, I wanted to look at the sensitivity of the PageRank vector for the $H200$ hyperlink graph. This theorem presented a formula for the derivative of the PageRank vector. To consider how the sensitivity varies in the range of $0 < \alpha < 1$, we can take the 1-norm of this resulting vector, $-v^T(I-S)(I-\alpha S)^{-2}$. To investigate this, I used Matlab to compute and plot the values of the norm for discrete values of α in the range $0.1 \leq \alpha \leq 0.9$, for the example hyperlink matrix $H200$. The

resulting graph is displayed in Figure 5 below.

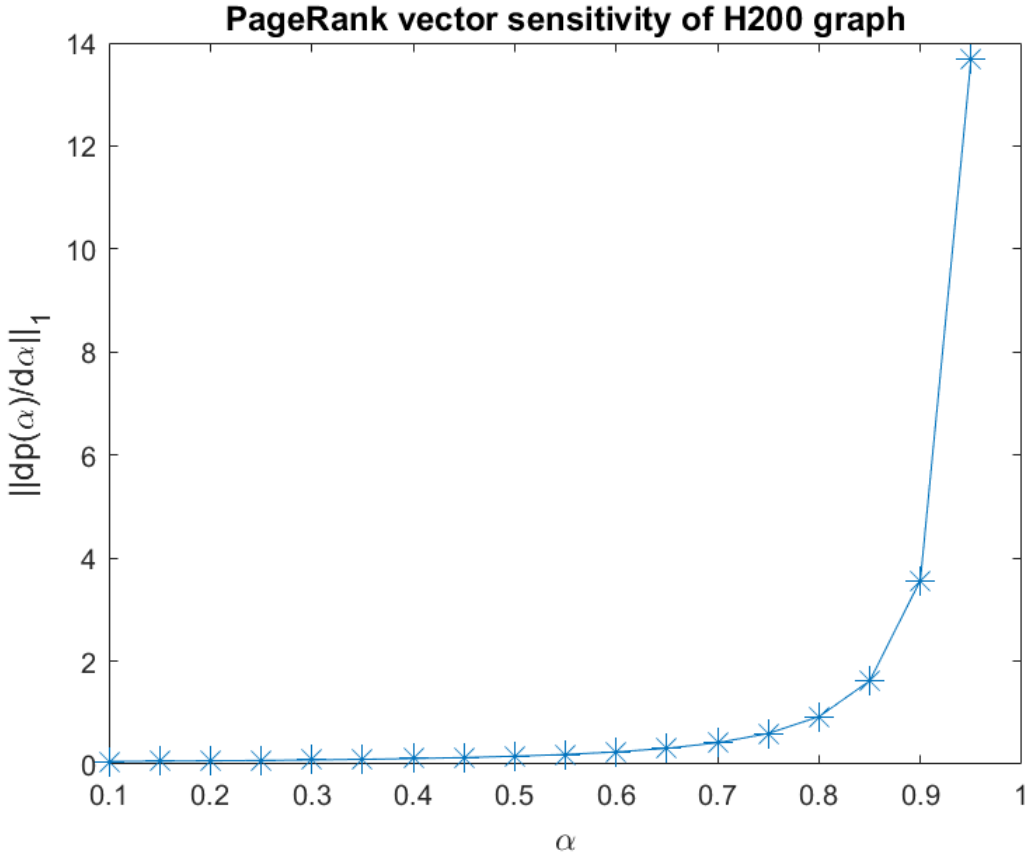

Figure 5

As expected, we can see that the PageRank vector is insensitive to small changes in α when α is small, but becomes increasingly sensitive to changes in α as α increases. In fact, as $\alpha \to 1$, the 1-norm blows up to ∞. This demonstrates the balancing act faced when determining an appropriate value of α to use in the PageRank model; with small α, PageRank values converge more quickly using the power method and will fluctuate much less, but on the other hand, using large values of α will more truly reflect the link structure of the Web.

7 Accelerating Computation

Recall that the rate of convergence of the power method can be approximated by the rate at which $\alpha^k \to 0$. What this means is that when the damping factor α is close to 1, the power method may perform poorly and require many iterations to obtain a

convergent PageRank vector. This slow convergence of the power method has lead to researchers investigating different ways of improving the method. The main reasons behind the need for acceleration are that:

- The Web is constantly changing and so PageRank values require regular updating.
- With the possibility of using different personalisation vectors, some newer approaches require the computation of multiple PageRank vectors.

Whilst the size and sparsity of a hyperlink matrix obtained from The Web make this method still dominate over other solutions, every iteration still comes at great cost due to the shear sizes involved, and so it is of clear importance to determine methods of reducing the number of iterations required to obtain high levels of accuracy.

7.1 Adaptive Power Methods

The following ideas for accelerating the rate of convergence of the power method are based on the observation that the rate of convergence of PageRank values for individual pages is non-uniform. This means that individual components of the PageRank vector may have converged to a predetermined degree of accuracy after, say, 30 iterations, but the power method will then blindly continue to carry out unnecessary calculations on these components until all components of the vector have converged. Because of this, an adaptive power method can be formed in which the PageRank of pages are not recomputed at each iteration once they have already converged to the required accuracy. Kamvar et al. [9] discuss exploiting this observation in this manner, and report an increase in computation speeds of 17% when carrying out large scale studies. The basic idea behind the adapted method is outlined below.

Assume we have carried out k iterations of the power method on the Google Matrix G and have obtained our current PageRank vector, $\pi^{(k)T}$. We wish to calculate the value at the next iteration, $\pi^{(k+1)T}$. We define C to be the set of webpages that have converged to a given degree of accuracy, and N to be those which have not. Then we can split our matrix G into 2 smaller matrices, the $(n \times m)$ matrix G_N corresponding to the in-going hyperlinks of those m pages whose PageRank values have not yet converged, and the $(n \times (n-m))$ matrix G_C corresponding to the in-going hyperlinks of the other $(n-m)$ pages.
Similarly, we can split our vector $\pi^{(k)T}$ into the vector $\pi_N^{(k)T}$ of length m corresponding to those values not yet converged, and the vector $\pi_C^{(k)T}$ of length $(n-m)$ corresponding to the values that have.
So we can then we order the matrix G and the PageRank vector $\pi^{(k)T}$ as follows:

$$G = \begin{pmatrix} G_N & G_C \end{pmatrix}, \pi^{(k)T} = \begin{pmatrix} \pi_N^{(k)T} & \pi_C^{(k)T} \end{pmatrix}. \tag{32}$$

Then the next iteration in the power method may be written as

$$\begin{pmatrix} \pi_N^{(k+1)T} & \pi_C^{(k+1)T} \end{pmatrix} = \begin{pmatrix} \pi_N^{(k)T} & \pi_C^{(k)T} \end{pmatrix} \cdot \begin{pmatrix} G_N & G_C \end{pmatrix}, \tag{33}$$

which results in the two iterative equations

$$\pi_N^{(k+1)T} = \pi_N^{(k)T} G_N \tag{34}$$

$$\pi_C^{(k+1)T} = \pi_C^{(k)T} G_C. \tag{35}$$

As the components of the vector $\pi_C^{(k)T}$ were assumed to have already converged after these k iterations, it is not necessary to compute the vector $\pi_C^{(k+1)T}$, and so equation (35) reduces to

$$\pi_C^{(k+1)T} = \pi_C^{(k)T}. \tag{36}$$

Identifying which web pages fall in the "converged" set C is inexpensive, although reordering the matrix G at each iteration is expensive [9]. Because of this, further adjustments and variations to the basic adaptive power method have been discussed with the aim of reducing computational cost. There are several problems with adaptive power methods such as these. For instance, though they have showed some positive results in numerical experiments, there is no theoretical proof regarding convergence of this method, and in some cases it would converge to incorrect final values [4]. This may be because, for example, the PageRank value of a page appears to remain constant for several iterations, and so the algorithm fixes this value, when in fact at the next iteration the value begins to change again.

7.2 Other acceleration methods

In addition to the adaptive power method discussed above, the same group of Stanford researchers have developed several further methods for acceleration. The first method, called **Aitken Extrapolation**, uses the assumption that the current iterate, $\pi^{(k)}$ in the power method can be expressed as a linear combination of the first two eigenvectors. By this assumption, one can solve for the principal eigenvector in closed form using the next two iterates, $\pi^{(k)}, \pi^{(k+1)}, \pi^{(k+2)}$ [10]. As we have seen that the number of power method iterations needed for a convergent PageRank vector depends on the absolute size of the subdominant eigenvalue, λ_2, the general idea of this method is as follows:

"If the subdominant eigenvalue causes the power method to sputter, cut it out and throw it away." [4]

Kamvar et al. found this method to have some success in decreasing the number of iterations required, but noticed it performed poorly in the situation that λ_2 and λ_3 were complex conjugates. Because of this, they developed an improved version of the method, namely **Quadratic Extrapolation**, which was more complex but based on

the same idea. In this method, the basic idea is that if λ_2 and λ_3 cause problems, they are both thrown away. When the power method was carried out using quadratic extrapolation on some test datasets, there was reported to be a $50 - 300\%$ reduction in computation time [4]. But there is still a compromise to be made, carrying out quadratic extrapolation at an iteration is expensive and so it may only be carried out every, say, 20 iterations. Thankfully, it turns out it does not need to be applied all that often to still achieve good results.
One final development of these ideas by this same group of researchers is an **Aggregation Method** called **BlockRank**. This method aims to reduce both number of iterations and work per iteration of the power method. In BlockRank, the webgraph is compressed into a **hostgraph**, in which nodes represent **hosts**, high-level webpages with many inlinks from other pages. There doesn't tend to be many links between said hosts, and those that do exist are ignored. The PageRank vector is computed independently for each host, and this is then multiplied by the probability of being in that host - called the expansion step. The probabiltity of being in a host, or equivalently the importance of this host, is determined by forming a host aggregation matrix [11]. Clearly we can see that the size of the PageRank vector for each host will be much smaller than a global PageRnk vector, and the size of the host aggregation matrix will be equal to the number of hosts. This method however only provides an approximation of the true PageRank vector, as some links in the webgraph have been ignored, but there are ways in which this approximation can be improved.
As well as these ideas, many other methods in extrapolation, aggregation, and numerical methods, have been researched and developed with the aim of accelerating the computation of the PageRank vector.

8 An Application in Football

8.1 Aims

I am going to investigate an application of PageRank in sports, namely its potential to rank the performance of individual players in a football match. Much of the research done in sports ranking using PageRank has been based around a 'winners network' in which nodes represent teams, and there is a directed arrow connecting team i and team j if team i is beaten by team j [12]. This has been widely investigated in the ranking of American Football teams, as well as for ranking tennis players. In these situations, the edges of the graph can be weighted by a quantity reflecting by how much team j won by, for example goal difference, or if teams play each other more than once then it may be the number of victories in a season [13]. Instead of using a ranking system based on winning games or matches, I will aim to rank the performance of individual players on the pitch, but many of the ideas are similar. The topic of graph theory has been seen to have great potential in this field, as by constructing a graph of passes made between players one is able to gain a greater understanding of a teams strategy. By

doing this it is then possible to use different centrality measures in order to determine the importance of individual players [14]. For my model, in order to investigate the way players of a team interact with each other throughout a game, I will construct a graph in which the nodes represent players, and node i links to node j if a successful pass is made from player i to player j. Furthermore, the arrows connecting nodes are weighted by the number of successful passes made. Once computed as a PageRank problem, each player will receive an overall ranking from the game, and roughly speaking, a player will receive a higher ranking if they receive more successful passes, coming from other 'good' players. What's more, this **Passing Network**, consisting of the players as nodes and passes as arrows, can be arranged such that the positions of the nodes relate to the positions of the players in terms of the team formation, and this should be able to provide insight into how the team play as a whole. Such insight may include any tactics and strategies the team may be using, and also on an individual level, if any player seems to be contributing more or less than their teammates.

8.2 Assumptions

In creating this model, there are several assumptions I will need to make and factors that need to be considered.

- I will consider the case in which a successful pass from player i to player j creates an entry in the ijth position in the link matrix, and so the idea of a 'good' player corresponds to the number of passes received - as opposed to the number made. Note this is in line with the way the original PageRank model was set out in section 2.1.

- It is important to recognise that ranking players based on how many successful passes they receive may not accurately reflect the performance of that player, as different positions require different skills from players. For example, it is probably going to be bias towards midfielders whose main role is to create passing play on the pitch, whereas clearly the main objective of a striker is to score goals. Despite of this, I am still going to include players of all positions in the model (except the goalkeeper) to see how much this bias is reflected in the model. In a more realistic model of footballers performances, many other factors, such as tackles made, fouls committed, shots on target, assists etc. would need to be considered.

- Another factor to consider is that over the course of the game, there will be substitutions made, and if a player only has 10 minutes of game time then clearly the number of passes they receive is likely to be much lower, even if they actually had great impact in the game in that short time. I have decided I will still include all players in the model (unless they played for a very short amount of time, in which case their passes will contribute to the player they replaced), but I will need to take this into account when comparing final rankings.

8.3 The Model

The game which I am going to study is a FIFA World Cup 2018 Semi-Final game, in which England played Croatia. I will attempt to model the England players performance using the Page Rank theory, and see how my results correspond to how these players were actually rated after the game by different sources.
The England team's starting line-up for this game, along with their positions, is shown in Table 2 below.

Table 2: Labelling of players in the game

Starting Line-Up		
Label	Position	Name
1	LCB	Harry Maguire
2	CB	John Stones
3	RCB	Kyle Walker
4	LWB	Ashley Young
5	CM	Jordan Henderson
6	RWB	Kieran Trippier
7	LM	Dele Alli
8	RM	Jesse Lingard
9	ST	Harry Kane
10	ST	Raheem Sterling
Substitutes		
11	ST	Marcus Rashford
12	CM	Eric Dier

I have assigned the players numerical labels as shown in the first column, which will later correspond to the player's row in the Google matrix. There were two substitutes made which are listed in the bottom section of the table, Rashford for Sterling, and Dier for Henderson. Two further substitutes were also made, Rose for Young, and Vardy for Walker, although in both of these cases the substitutes only came on in extra-time and so there was not much time at all to collect data for these players. Because of this, I made the decision to just include the few passes that they did make/receive into the row/column of the player they came on for.
By watching the game, i recorded all successful passes between players and entered them into the following table, i.e. I added +1 in the ij^{th} entry of the table for every successful pass made from Player i to Player j. This gave the following table:

Table 3: Number of successful passes between players during the match

Player	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**Total**
1	0	8	4	8	3	1	4	2	2	0	1	3	36
2	8	0	14	4	4	3	4	3	0	1	2	4	47
3	4	10	0	1	8	8	1	4	1	0	0	2	39
4	5	0	1	0	0	0	5	2	0	0	3	0	16
5	3	3	8	1	0	4	4	4	1	1	0	0	29
6	0	0	8	0	5	0	0	5	4	1	3	0	26
7	5	3	1	5	2	1	0	2	3	6	1	0	29
8	4	3	2	0	5	9	2	0	3	3	2	2	35
9	0	0	0	1	2	3	6	1	0	3	1	0	17
10	0	0	0	0	2	0	2	2	1	0	0	0	7
11	0	0	1	0	0	2	0	2	1	0	0	1	7
12	2	6	2	0	0	2	0	3	0	0	0	0	15

From this data, I created a directed graph, showing the starting 10 players, numbered as set out above, each representing a node. In this graph the nodes are arranged to represent the formation of the squad, and directed arrows from one node to another indicate that at least one pass has been made from one player to another. In addition, I have weighted the thickness of the directed arrows to correspond to the number of passes made from one player to the next, in order to display visually where on the pitch most passes were made throughout the game. Note that for simplicity this graph only represents the actions of the players $1-10$, and not the two substitutes 11 and 12. This directed graph is shown in Figure 6 below.

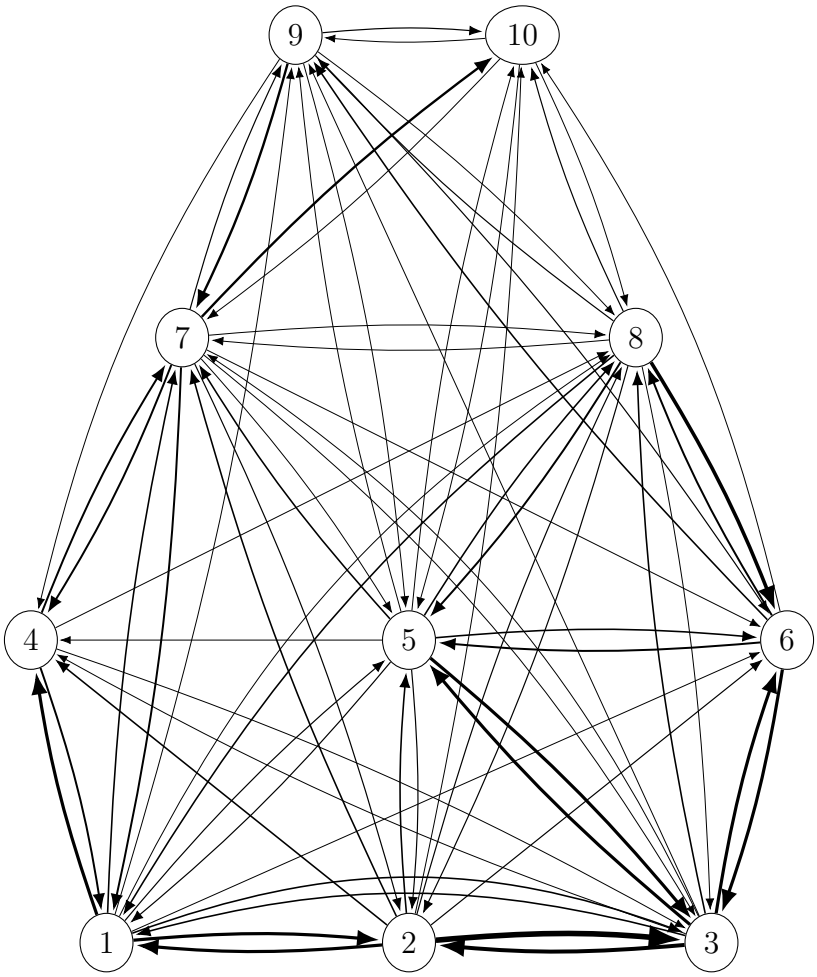

Figure 6: England Team Passing Network

Now the first step in creating the 'Google matrix' for this system was to create the row stochastic matrix S. We denote the totals calculated in the right-most column of the above table as $|P_i|$, the total number of successful passes made by player i for each $1 \leq i \leq 12$. The we can obtain the row normalised matrix S, which has $S_{ij} = \frac{P_{ij}}{|P_i|}$ where there are P_{ij} passes from player i to j, and 0 otherwise. The non-zero elements of row i correspond to the successful passes made by player i, whilst the non-zero elements of column i correspond to the successful passes received by player i. We note that in this example, a zero-row would correspond to a player not making any successful passes over the course of a game, which clearly will not exist in this scenario. Therefore, our matrix is already fully stochastic and we have no need to make the stochasticity adjustment as discussed in subsection 3.1.
Therefore, our matrix S is given by:

$$S = \begin{pmatrix} 0 & \frac{8}{36} & \frac{4}{36} & \frac{8}{36} & \frac{3}{36} & \frac{1}{36} & \frac{4}{36} & \frac{2}{36} & \frac{2}{36} & 0 & \frac{1}{36} & \frac{3}{36} \\ \frac{8}{47} & 0 & \frac{14}{47} & \frac{4}{47} & \frac{4}{47} & \frac{3}{47} & \frac{4}{47} & \frac{3}{47} & 0 & \frac{1}{47} & \frac{2}{47} & \frac{4}{47} \\ \frac{4}{39} & \frac{10}{39} & 0 & \frac{1}{39} & \frac{8}{39} & \frac{8}{39} & \frac{1}{39} & \frac{4}{39} & \frac{1}{39} & 0 & 0 & \frac{2}{39} \\ \frac{5}{16} & 0 & \frac{1}{16} & 0 & 0 & 0 & \frac{5}{16} & \frac{2}{16} & 0 & 0 & \frac{3}{16} & 0 \\ \frac{3}{29} & \frac{3}{29} & \frac{8}{29} & \frac{1}{29} & 0 & \frac{4}{29} & \frac{4}{29} & \frac{4}{29} & \frac{1}{29} & \frac{1}{29} & 0 & 0 \\ 0 & 0 & \frac{8}{26} & 0 & \frac{5}{26} & 0 & 0 & \frac{5}{26} & \frac{4}{26} & \frac{1}{26} & \frac{3}{26} & 0 \\ \frac{5}{29} & \frac{3}{29} & \frac{1}{29} & \frac{5}{29} & \frac{2}{29} & \frac{1}{29} & 0 & \frac{2}{29} & \frac{3}{29} & \frac{6}{29} & \frac{1}{29} & 0 \\ \frac{4}{35} & \frac{3}{35} & \frac{2}{35} & 0 & \frac{5}{35} & \frac{9}{35} & \frac{2}{35} & 0 & \frac{3}{35} & \frac{3}{35} & \frac{2}{35} & \frac{2}{35} \\ 0 & 0 & 0 & \frac{1}{17} & \frac{2}{17} & \frac{3}{17} & \frac{6}{17} & \frac{1}{17} & 0 & \frac{3}{17} & \frac{1}{17} & 0 \\ 0 & 0 & 0 & 0 & \frac{2}{7} & 0 & \frac{2}{7} & \frac{2}{7} & \frac{1}{7} & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{7} & 0 & 0 & \frac{2}{7} & 0 & \frac{2}{7} & \frac{1}{7} & 0 & 0 & \frac{1}{7} \\ \frac{2}{15} & \frac{6}{15} & \frac{2}{15} & 0 & 0 & \frac{2}{15} & 0 & \frac{3}{15} & 0 & 0 & 0 & 0 \end{pmatrix}. \tag{37}$$

Next we recall the primitivity adjustment discussed in section 3.2, in which we form the convex combination of the matrix S and the teleportation matrix given by $\frac{1}{n}ee^T$, in order to create a positive, and hence primitive matix G. Taking the standard value of α to be 0.85, we have

$$G = \alpha S + (1-\alpha)\frac{1}{n}ee^T = 0.85S + 0.15 \times \frac{1}{12} \times ee^T$$

$$= \begin{pmatrix} \frac{1}{80} & \frac{29}{144} & \frac{77}{720} & \frac{29}{144} & \frac{1}{12} & \frac{13}{360} & \frac{77}{720} & \frac{43}{720} & \frac{43}{720} & \frac{1}{80} & \frac{13}{360} & \frac{1}{12} \\ \frac{591}{3760} & \frac{1}{80} & \frac{999}{3760} & \frac{319}{3760} & \frac{319}{3760} & \frac{251}{3760} & \frac{319}{3760} & \frac{251}{3760} & \frac{1}{80} & \frac{23}{752} & \frac{183}{3760} & \frac{319}{3760} \\ \frac{311}{3120} & \frac{719}{3120} & \frac{1}{80} & \frac{107}{3120} & \frac{583}{3120} & \frac{583}{3120} & \frac{107}{3120} & \frac{311}{3120} & \frac{107}{3120} & \frac{1}{80} & \frac{1}{80} & \frac{35}{624} \\ \frac{89}{320} & \frac{1}{80} & \frac{21}{320} & \frac{1}{80} & \frac{1}{80} & \frac{1}{80} & \frac{89}{320} & \frac{19}{160} & \frac{1}{80} & \frac{1}{80} & \frac{11}{64} & \frac{1}{80} \\ \frac{233}{2320} & \frac{233}{2320} & \frac{573}{2320} & \frac{97}{2320} & \frac{1}{80} & \frac{301}{2320} & \frac{301}{2320} & \frac{301}{2320} & \frac{97}{2320} & \frac{97}{2320} & \frac{1}{80} & \frac{1}{80} \\ \frac{1}{80} & \frac{1}{80} & \frac{57}{208} & \frac{1}{80} & \frac{183}{1040} & \frac{1}{80} & \frac{1}{80} & \frac{183}{1040} & \frac{149}{1040} & \frac{47}{1040} & \frac{23}{208} & \frac{1}{80} \\ \frac{369}{2320} & \frac{233}{2320} & \frac{97}{2320} & \frac{369}{2320} & \frac{33}{464} & \frac{97}{2320} & \frac{1}{80} & \frac{33}{464} & \frac{233}{2320} & \frac{437}{2320} & \frac{97}{2320} & \frac{1}{80} \\ \frac{307}{2800} & \frac{239}{2800} & \frac{171}{2800} & \frac{1}{80} & \frac{15}{112} & \frac{647}{2800} & \frac{171}{2800} & \frac{1}{80} & \frac{239}{2800} & \frac{239}{2800} & \frac{171}{2800} & \frac{171}{2800} \\ \frac{1}{80} & \frac{1}{80} & \frac{1}{80} & \frac{1}{16} & \frac{9}{80} & \frac{13}{80} & \frac{5}{16} & \frac{1}{16} & \frac{1}{80} & \frac{13}{80} & \frac{1}{16} & \frac{1}{80} \\ \frac{1}{80} & \frac{1}{80} & \frac{1}{80} & \frac{1}{80} & \frac{143}{560} & \frac{1}{80} & \frac{143}{560} & \frac{143}{560} & \frac{15}{112} & \frac{1}{80} & \frac{1}{80} & \frac{1}{80} \\ \frac{1}{80} & \frac{1}{80} & \frac{15}{112} & \frac{1}{80} & \frac{1}{80} & \frac{143}{560} & \frac{1}{80} & \frac{143}{560} & \frac{15}{112} & \frac{1}{80} & \frac{1}{80} & \frac{15}{112} \\ \frac{151}{1200} & \frac{141}{400} & \frac{151}{1200} & \frac{1}{80} & \frac{1}{80} & \frac{151}{1200} & \frac{1}{80} & \frac{73}{400} & \frac{1}{80} & \frac{1}{80} & \frac{1}{80} & \frac{1}{80} \end{pmatrix}. \tag{38}$$

Finding the stationary vector of G gives

$$\pi^T = \begin{pmatrix} 0.09212 & 0.09659 & 0.12040 & 0.06053 & 0.10373 & 0.10658 & 0.09802 & 0.10703 & 0.06608 \\ & 0.05606 & 0.04998 & 0.04286 & & & & & \end{pmatrix}. \tag{39}$$

This gives that, for example, player 3 had had most importance in the game, and 12 had least importance. The ranking that I have determined is listed in the following table.

Table 4: Ranking of players according to the PageRank model

Position	**Player**
1	Walker
2	Lingard
3	Tripper
4	Henderson
5	Alli
6	Stones
7	Maguire
8	Kane
9	Young
10	Sterling
11	Rashford
12	Dier

Next I collected several different player ratings from different popular and well regarded sports channels. The points awarded for each player (out of 10) for their performance in the match is displayed in the following table.

Table 5: Player ratings found online by different sports channels

Player	**BBC**	**BBC Player Rating**	**Sky Sports**	**The Guardian**	**ESPN**
Walker	6	6.22	8	7	8
Lingard	5	5.84	6	6	6
Trippier	8	7.47	8	7	8
Henderson	6	6	7	7	6
Alli	5	5.37	6	7	5
Stones	7	6.29	6	6	5
Maguire	7	6.55	7	8	7
Kane	5	5.11	6	6	5
Young	5	5.3	6	6	6
Sterling	6	5.59	7	6	7
Rashford	6	5.54	6	6	5
Dier	5	5.24	5	6	5

Note that each source did not necessarily provide a ranking in the way I was hoping to obtain, as often there are multiple players that received the same score. In order to compare these scores with my own ranking of the players, I next computed the normalised scores for each of these scores. That is, for each column I summed the total score from

all players and then divided each individual player's score by this total. I then took an average of the different normalised scores for each player, so I could use these averages to determine a new ranking for the 12 players. As each of the ratings given by the different channels vary, due to the fact they are formulated by different methods or are based on different sports writer's opinions, I decided to use an average of their scores to give a collective opinion and a better overall view of 'which player was the best'. This data is displayed in Table 6 below.

Table 6: Normalised player ratings from different sports channels

Player	**BBC**	**BBC Player Rating**	**Sky Sports**	**The Guardian**	**ESPN**	**Average**
Walker	0.08451	0.0882	0.10256	0.08974	0.10959	0.09492
Lingard	0.07142	0.08281	0.07692	0.07692	0.08219	0.07805
Trippier	0.11268	0.10593	0.10256	0.08974	0.10959	0.1041
Henderson	0.08451	0.08508	0.08974	0.08974	0.08219	0.08625
Alli	0.07042	0.07615	0.07962	0.08974	0.06849	0.07688
Stones	0.09859	0.08919	0.07692	0.07692	0.06849	0.08202
Maguire	0.09859	0.09288	0.08974	0.10256	0.09589	0.09593
Kane	0.07042	0.07246	0.07692	0.07692	0.06849	0.07304
Young	0.07042	0.07516	0.07692	0.07692	0.08219	0.07632
Sterling	0.08451	0.07927	0.08974	0.07692	0.09589	0.08527
Rashford	0.08451	0.07856	0.07692	0.07692	0.06849	0.07708
Dier	0.07042	0.07431	0.0641	0.07692	0.06849	0.07085

Using the averages in the right-hand column, I could determine a new ranking for the players based on the average points they were awarded by these 5 sports channels. This new ranking is displayed in Table 7 below.

Finally, I created a scatter graph displaying the positions of the players I determined using the PageRank model, marked by crosses, compared to the position of these players determined by the average scores of the 5 different sports channels, marked by circles. The vertical lines indicate by how many positions each player differs according to the two different rankings. This scatter graph is displayed in Figure 7.

Table 7: Position of player according to new ranking system

Player	Position
Walker	3
Lingard	7
Trippier	1
Henderson	4
Alli	9
Stones	6
Maguire	2
Kane	11
Young	10
Sterling	5
Rashford	8
Dier	12

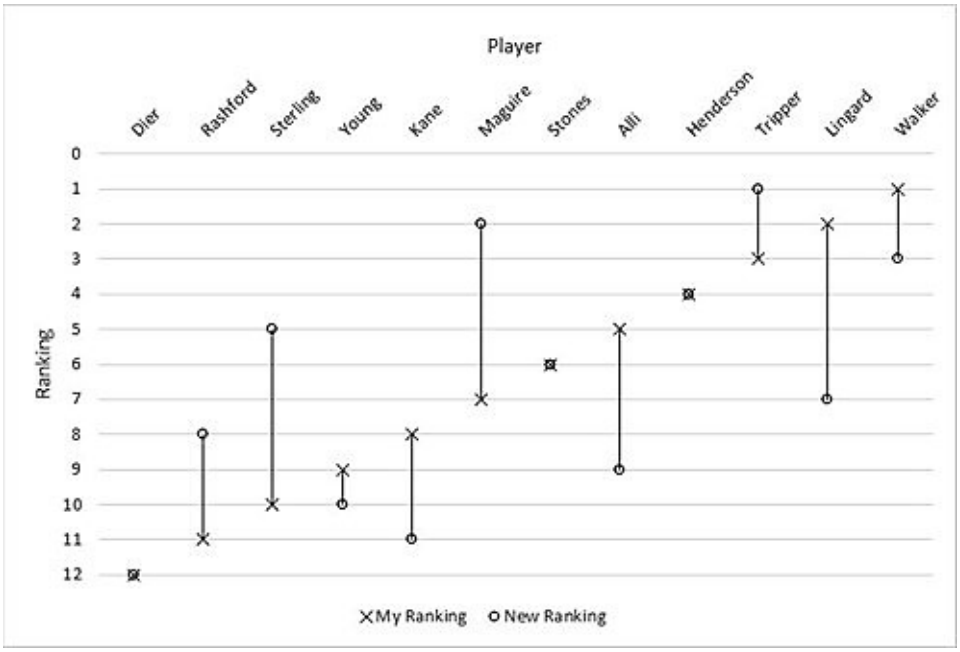

Figure 7: Scatter Plot to show the difference in players' positions across the two rankings

8.4 Analysis

Studying the graph I produced in Figure 7 above, there is clearly a correlation between the ranking produced using the PageRank model and the one produced using a range of sports writers' opinions. In fact, you can see that 3 players held the same position in both rankings, and I calculated that as an average, a player differed by 2.5 positions across the both rankings. This was surprisingly close considering the PageRank theory I

applied in this context created a very simplistic model, and so clearly the idea of ranking players by how many passes they received and whom they received them from actually creates a reasonably accurate model for how well the player performed. The players that differed the most between the two rankings were Sterling, Maguire and Lingard. In the case of centre-back Maguire, for example, the PageRank model put him mid-table in the rankings, whilst he was generally regarded to have been one of the best on the pitch. This could be because, for example, in my model he didn't score too highly as the amount of passes he received was fairly average in the team. On the other hand, Maguire is a defender and made many vital tackles in the game which he was highly praised for, but these would not be reflected in my passing model.

In order to make improvements to my model, I think I would need to develop a more complex system by which points were awarded to players for all types of behaviour on the pitch, not just for passing alone. For example, points could be awarded for tackles made, shots on target etc. so that the model better suits players of all positions. Furthermore, points could be deducted for foul play during the game. I would probably also analyse multiple games and use this to create a general rating for the player, not just their performance specifically in one game. This is primarily to increase the amount of data collected, as recording just the amount of passes a player received in one game does not create a very large set of data. In addition, I would be able to compare my results with the offical FIFA player ratings, which combine six scores for key attributes; pace, shooting, passing, dribbling, defending, and physical, and combine these with a player's international recognition to calculate the player's overall rating [15]. As FIFA uses this much more complex rating system, and many of the sports channel's ratings I looked at are largely based on one individual's opinion, this would be a much better source of reference when comparing my results. Regardless, I have demonstrated an application of PageRank theory in the field of sporting performance that has lead to a reasonably accurate ranking.

9 Other Applications

9.1 Road Networks

Another application of the PageRank theory I am going to discuss is in modelling road networks and urban spaces. By creating a network graph in which roads are linked in the natural way, i.e. if two roads are adjacent, one can implement the PageRank theory to help predict both human and traffic flow in urban areas [12]. This can be done on a small scale, for example modelling traffic flow in and around a city centre, or on a larger scale, using particular urban spaces as hubs and looking at the movement between these areas. An urban space in this context is generally thought of as a neighborhood or block in a city. Two urban spaces are joined in the network if they are physically adjacent. The extended and weighted PageRank algorithms have been used to model human movement

in four areas of London, and a significant correlation was discovered between the observed data and the values determined using the PageRank evaluation [16]. In this study, the best results were obtained when using weighted PageRank with $\alpha = 1$. The **Weighted PageRank** algorithm is an extension of standard PageRank in which larger PageRank values are assigned to those pages viewed to be more 'popular'. Instead of assigning the rank value of a page evenly between the out-linking pages, each out-linking page receives a value that is proportional to the number of links entering and leaving the page [17]. To visualise this, consider the connectivity graph below. In this context, this edges of this graph may be thought of as the links between urban spaces (how ever these are chosen to be defined), which represent the nodes.

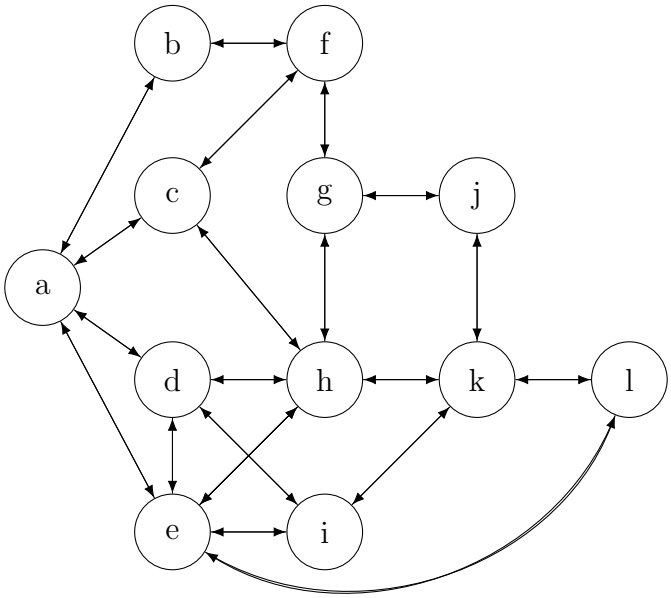

Figure 8: Connectivity graph of urban spaces

One can imagine they are a pedestrian standing at point a. Under the standard PageRank algorithm, it is assumed there is an equal probability of $\frac{1}{4}$ of taking each of the routes to the points b, c, d and e. On the other hand, using a weighted PageRank, the pedestrian is most likely to next travel to point e, as this location has the highest connectivity out of the 4 adjacent points. In this case, the probability of next visiting the points b, c, d and e, respectively, would be $\frac{2}{14}$, $\frac{3}{14}$, $\frac{4}{14}$ and $\frac{5}{14}$, where the 14 represents the total number of links of the 4 pages.
So using the weighted PageRank here with $\alpha = 1$ represents the situation in which 'important' urban spaces are likely to be close to other 'important' urban spaces, and the teleportation matrix (see Section 5.3) is eliminated entirely from the Google matrix. Intuitively, this makes sense because individuals and businesses are likely to co-locate areas with high connectivity, and individuals cannot teleport over the short time-frames

used in these kind of experiments. If we were to look at a larger time-scale, it may be the case that using a value of $\alpha < 1$ would better represent the situation [12].
The study of PageRank's use in modelling flow of traffic and pedestrians in urban environments has lead to many wider applications. For example, it provides insight into consumer behaviour, can assist in the planning of effective public transport systems, route planning, and also in the management of traffic lights. What's more, researchers have studied the use of PageRank theory to identify areas of particularly high terrorism risk, as transportation hubs and busy urban areas are often targeted [18].

9.2 Ecology

There has also been a lot of research into the use of PageRank across many different parts of science, including chemistry, physics, biology and bioinformatics, and neuroscience. One final application I am going to briefly discuss is in ecology, where PageRank has a potential use in determining which species are the most important in an ecosystem.
Food webs are complex networks representing ecosystems. They display the relationships existing between different species and most importantly, 'who eats whom'. Identifying which species are of most importance is vital in the study of ecosystems, as it may be that if these species were to go extinct, it would cause a large succession of extinctions further up the food chain, causing a collapse of the system. Whilst the notion of this chain-reaction of extinctions has been understood by scientists for a long time, the matter of determining which species are of greatest important is not trivial, due to the vast number of interactions between different plant and animal species.
Researchers realised PageRank could be applied to this problem [19], but a couple of adjustments had to be made. First, the algorithm must be 'reversed' for it to make sense in this context. To see why, recall that PageRank gives that a web page is more important if it is linked to by other important pages. Similarly, as set out in Section 8, a 'good' player corresponded to one that received passes from other good players. The difference here is that a species in the ecosystem should be seen as important if it points to other important species, as this means these other important species are relying on it as a food supply for survival. Another adjustment is that the network of species needs to be more cyclic in order for the algorithm to be applicable, otherwise species at the top of food chains will point to no other species - relating to the issue of dangling nodes. For example, not much eats a lion, but this does not mean a lion is not an important part of the food chain. Mathematically speaking, food webs are not irreducible nor primitive, which we know are properties needed for convergence of the PageRank algorithm. In this context, introducing a damping factor makes no sense, as nutrients cannot randomly jump between species. Here, overcoming this problem was done by including a so-called 'detritus pool', which relates to the idea that when an organism produces excrement, or dies and decomposes, it provides energy for the plants, which in turn feed organisms at the bottom of the food web, and so on. This is added in the network via a 'root' node, which points to all primary producers, and is pointed to by all living organisms in the

system. This observation turns the food web system into a cyclic system which is now irreducible and primitive, and allows the reversed PageRank algorithm to be applied. In Figure 9 is an simple example of a food web, containing 7 species labelled $a - f$, and the root node. Here you can see that all species point back to the root node via the dashed arrows.

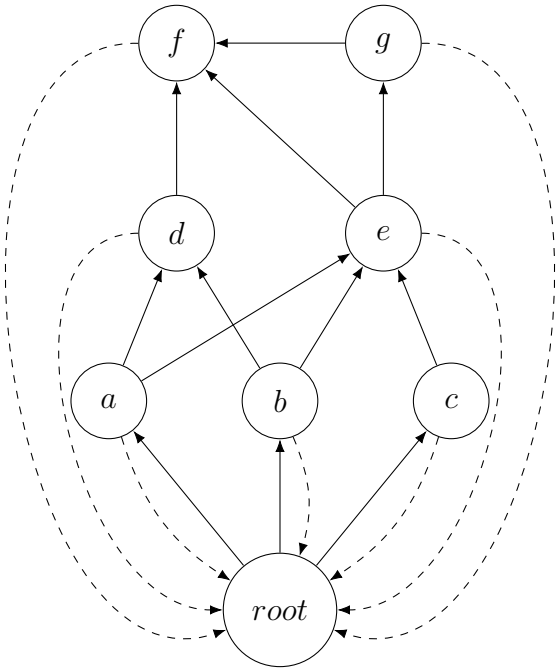

Figure 9: An example food web

In their 2009 study [19], Allesina and Pascual used the algorithm to rank species in terms of how much damage they would do to the food web system should they go extinct, and found that for 12 published food webs PageRank actually outperformed the more complex algorithms already in use in computational biology, in terms of 'fastest route to collapse' in the food web.

This final example of an application of PageRank theory demonstrates further the importance of such mathematical theory, and how one simple idea put forward in 1996 by computer-science students Brin and Page can still be of great relevance to us today across a wide range of applications.

10 Appendix

10.1 Theorems

Theorem 10.1.1 *If $\rho(M) < 1$ for a matrix M then the matrix $I - M$ is invertible.*

Proof. Suppose $I - M$ is not invertible, then it has a non-trivial kernel and so $\exists v \neq 0$ such that $(I - M)v = 0$, i.e. $Mv = v$. This shows that 1 is an eigenvalue of M, hence $1 \in \sigma(M)$. Therefore the spectral radius of M, $\rho(M)$, is greater than or equal to 1. This proves the result. □

Theorem 10.1.2 *If $X \in \mathbb{R}^{n\times n}$ is such that $||X|| < 1$ then $I - X$ is non-singular and we have that*

$$(I - X)^{-1} = \sum_{j=0}^{\infty} X^j \geq 0.$$

Proof. First note that

$$(I - X)\sum_{j=0}^{k} X^j = \sum_{j=0}^{k} X^j - \sum_{j=1}^{k+1} X^j = I - X^{k+1}.$$

Considering this in the case $k \to \infty$, we obtain

$$(I - X)\sum_{j=0}^{\infty} X^j = \lim_{k\to\infty}(I - X)\sum_{j=0}^{k} X^j = \lim_{k\to\infty} I - X^{k+1} = I, \tag{40}$$

noticing that

$$\lim_{k\to\infty} ||X^k - 0|| \leq \lim_{k\to\infty} ||X||^k = 0.$$

Rearranging (40) we see that

$$(I - X)^{-1} = \sum_{j=0}^{\infty} X^j,$$

and to show this summation is non-negative simply notice that if $X \geq 0$ then $X^j \geq 0$ and so $\sum_{j=0}^{\infty} X^j \geq 0$. □

Theorem 10.1.3 *For an invertible matrix A, we have*

$$\frac{dA(\alpha)^{-1}}{d\alpha} = -A^{-1}(\alpha)\frac{dA(\alpha)}{d\alpha}A^{-1}(\alpha).$$

Proof. Note that

$$0 = (I_n)' = (A(\alpha)A^{-1}(\alpha))' = A(\alpha)'A^{-1}(\alpha) + A(\alpha)(A^{-1}(\alpha))',$$

using the chain rule. Then,

$$A(\alpha)(A^{-1}(\alpha))' = -A(\alpha)'A^{-1}(\alpha) \implies A^{-1}(\alpha))' = -A^{-1}(\alpha)A(\alpha)'A^{-1}(\alpha).$$

□

10.2 Matlab Code

Forming the stochastic matrix S for the hyperlink matrix $H = H200$

```
> H = H200;
> n = length(H);
> c = full(sum(H));
> k = find(c ~= 0);
> D = sparse(k,k,1./c(k),n,n);
> a = sparse(c == 0)/n;
> S = H*D + a(ones(n,1),:);
```

Carrying out the power method for $\alpha = 0.85$ and a tolerance of 10^{-5}

```
> α = 0.85;
> v = ones(n,1)/n;
> u = (1-α)*v;
> p = ones(n,1)/n;
> p_prev = zeros(n,1);
> cnt = 0;
> while max(abs(p-p_prev)) > 1e-5
  p_prev = p;
  p = alpha*S*p + u;
  cnt = cnt + 1;

> end
```

Verification that $||\lambda_2(G)|| \approx \alpha$ for the hyperlink matrix $H200$ and $\alpha = 0.85$

After forming S as above, I then used the following code to find the eigenvalues of the Google matrix G:

$> \ \alpha = 0.85 \ ;$

$> \ T = (1/n) * ones(n, n) \ ;$

$> \ G = \alpha * S + (1 - \alpha) * T \ ;$

$> \ e = eig(G)$

Returns the vector $e = (1.0000, 0.8387, 0.6454, 0.5592, 0.4297, 0.3132, \dots)$, and from this we see that $||\lambda_2(G)|| = 0.8387 \approx 0.85$.

References

[1] https://searchengineland.com/google-now-handles-2-999-trillion-searches-per-year-250247

[2] https://www.worldwidewebsize.com/

[3] S.Brin and L.Page. *The Anatomy of a Large-Scale Hypertextual Web Search Engine*, Pages 101-117, Computer Networks vol. 30, Computer Science Department, Stanford University, Stanford, CA 94305, 1998.

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

[5] L.Page, S.Brin, R.Motwani and T.Winograd. *The PageRank Citation Ranking: Bringing Order to the Web*, Stanford University, Stanford, CA 94305, 1999.

[6] https://ergodic.ugr.es/cphys/LECCIONES/FORTRAN/power_method.pdf

[7] http://mathworld.wolfram.com/ReducibleMatrix.html

[8] C.D.Meyer. *Matrix analysis and applied linear algebra*, SIAM, Philadelphia, PA, USA, 2000.

[9] S.Kamvar, T.Haveliwala and G.Golub. *Adaptive Methods for the Computation of PageRank*, Pages 51-65, Linear Algebra and its Applications, Stanford University Stanford, CA 94305, 2004.

[10] S.Kamvar, T.Haveliwala, C.Manning and G.Golub. *Extrapolation Methods for Accelerating PageRank Computations*, Pages 261–270, WWW '03: Proceedings of the 12th international conference on World Wide Web, Stanford University Stanford, CA 94305, 2003.

[11] A.N.Langville and C.D.Meyer. *Deeper Inside PageRank*, Pages 335-380, Internet Mathematics, 2004.

[12] D.F.Gleich. *PageRank Beyond The Web*, SIAM Rev., 57(3), 321–363, 2015.

[13] J.P.Keener. *The Perron-Frobenius Theorem and the Ranking of Football Teams*, SIAM Rev., 35(1), Pages 80–93, 1993.

[14] V.Lazova and L.Baznarkov. *PageRank Approach to Ranking National Football Teams*, Faculty of Computer Science and Engineering, Cyril and Methodius University, Skopje, R.Macedonia, 2015.

[15] https://www.goal.com/en-gb/news/fifa-player-ratings-explained-how-are-the-card-number-stats/1hszd2fgr7wgf1n2b2yjdpgynu

[16] B.Jiang. *Ranking spaces for predicting human movement in an urban environment*, Pages 823-837, International Journal of Geographical Information Science, 2009.

[17] W.Xing and A.Ghorbani. *Weighted PageRank Algorithm*, Faculty of Computer Science, University of New Brunswick Fredericton, NB, E3B 5A3, Canada.

[18] J.Chan and K.Teknomo. *Hub Identification of the Metro Manila Road Network Using PageRank*, School of Science and Engineering, Ateneo de Manila University, Philippines 1108, 2015.

[19] Stefano Allesina and Mercedes Pascual. *Googling Food Webs: Can an Eigenvector Measure Species' Importance for Coextinctions?*, PLoS Computational Biology, 2009.