21-241 Term Project: The PageRank Algorithm

Ishika Saxena and Eesha Sharma

December 2019

1 Introduction

PageRank, invented by Larry Page and Sergei Brin of Stanford University in 1996, is an algorithm that is used in various applications throughout the world of tech, perhaps most famously in Google's web-searching system. In this paper, we explore the core fundamental ideas of the algorithm.

PageRank's application is heavily based on the Markov Chain, a specific type of stochastic system which defines the transition of a set of random variables from one state to another in accordance with certain probabilistic tendencies/rules. Further, in this system, the probability of transitioning to a particular state is entirely dependent on the current state (rather than the preceding sequence of states). Markov Chains have significant applications in mathematics, finance, linguistics, and a plethora of other subjects by virtue of their ability to outline the probabilities of systems transitioning between states. In PageRank, we see Markov Chains applied in the context of linear algebra via Markov adjacency matrices.

It is this application of Markov chains that perhaps made PageRank so prominent and made Google the primary search engine rather than its predecessors like Alta Vista, which used classic information retrieval techniques based primarily on processing the text of pages rather than the links inside them to rank web pages, which, over time, caused spam to persist and low quality results to prevail (Broadley).

2 Relevant Definitions and Theorems

Directed Graph, Web Graph
A directed graph is a collection of nodes and edges where the nodes are the objects and the edges are the connections between these objects. Unlike undirected graphs, the edges of directed graphs do not indicate a two-way relationship. A web graph is a specific type of directed graph, in which the nodes are the web pages and the edges are the hyperlinks between the web pages. Essentially, the static Web can be viewed as a web/directed graph of various HTML pages,

where a hyperlink on a page represents that page linking to an external page.

Adjacency Matrix for a Directed Graph
The adjacency matrix A for a directed graph is an n x n matrix (where n is the number of nodes on the graph) where Aij = 1 if there is an edge going from node j to node i and Aij = 0 otherwise.

Stochastic Matrix (also called Transition Matrix and Markov Matrix)
A square $n \times n$ matrix A is a stochastic, or transition, matrix if it describes the transitions from one state to another according to certain probabilistic tendencies. (Essentially, a stochastic matrix describes the transitions of a Markov chain, which is defined below.) In such a matrix, the A_{ij} entry of matrix A (the entry in row i, column j of matrix A) describes the probability of some input transitioning from its current state j to state i. In accordance with this, the A_{ij} entry can also be interpreted as the conditional probability that "we are on the ith page at step n+1, given that we were on the jth page at step n" (Rousseau). Since the total transition probability from state j to all other states must equal 1 (as all the probabilities in a sample space are defined to sum to 1), the columns of a stochastic matrix sum to 1. Each entry in a stochastic matrix is non-negative (i.e. greater than or equal to zero). Because a stochastic matrix satisfies these two aforementioned properties, stochastic matrices are also Markov matrices.

Random Walk on a Directed Graph
A random walk is a process in which a variable takes on completely random values. An example relating to a directed graph could be a person walking from node to node on a graph, in which at each time step, the person chooses randomly which path to follow to another node (where each possible edge the person can use to move has equal probability). The concept of random walk can be extended to directed web graphs, in which the user could proceed from some page A to any of the pages (B, C, D) that A links to with equal probability. A user's random walk essentially can nicely be displayed as an adjacency matrix and interpreted in terms of a transition matrix of probabilities of transitioning from one state to another.

Steady State Vector
The steady state vector of stochastic matrix A is the vector that, when multiplied by A, results in itself. In other words, x is a steady state vector if Ax = x. By the definition of eigenvalues and eigenvectors, we see that the steady state vector x is actually the eigenvector corresponding to the eigenvalue of 1 of matrix A.
Beyond the property of Ax = x, we also see that upon repeated multiplication

of an initial vector x0 by M, we obtain a string of equations like this:

$$x_k + 1 = Ax_k \text{ and } x_k + 2 = Ax_k + 1$$

i.e.

$$x_k + 2 = A^2 x_k$$

This leads us to the equation $A^k * x0 = x$. (Note that this is also stated in Power Method Convergence Theorem below.)
This is in alignment with our explanation of transition matrices of probability, as the steady state vector shows us that equilibrium is reached (i.e. the probabilities eventually converge) irrespective of the starting state via repeated matrix multiplication by A. The steady state vector can be normalized by having its components sum to 1 so that the values clearly depict probability. And in the case of a web graph, the steady state vector displays the ranks of the pages, because it shows how the probabilities of the random walker transitioning from one state to another converges, in that there is a location(s) where the random walker is more likely to be than the other. In a web graph, the steady state vector is often called the pagerank vector.

Markov Chain
A Markov Chain describes the transition of a set of random variables from one state to another in accordance with certain probabilistic tendencies/rules. This set of transitions satisfies the Markov Property, which states that the probability of transitioning to a particular state is entirely dependent on the current state (rather than the preceding sequence of states). In relation to linear algebra, we see how Markov Chains allow us to construct the steady state vector of a square stochastic matrix A via an iterative process, in which a vector x_0 is inputted and multiplied by M and, thereupon, each subsequent prediction of $x_n + 1$ is made by matrix-vector multiplication Mx_n.
Eventually, a point is reached where $x_k + 1 = Ax_k$, which allows us to write $A^k * x_0$, allowing us to see that these expressions give us the steady state (Note that this is also stated in Power Method Convergence Theorem below.) Thus, Markov Chains essentially allow us to have a sequence of vectors $x_0, x_1, x_2, \ldots, x_n$such that there exists a stochastic matrix A s.t. $Ax_k = x_k + 1$.

Rate of Convergence
As explained in the two above terms, the steady state vector describes the convergence of probabilities in relation to one state transitioning to another. By this, we obtain $A^k * x_0 = x$. In other words, for some initial n x 1 column vector x, x0 is repeatedly replaced with the product Ax where A is an n x n matrix until x0 converges to x, the steady state, or pagerank, vector.
In terms of directed graphs and convergence, we see that A would represent the directed graph and a random walker would essentially pick an edge randomly

from each node and go to the selected page. After some amount of time, the n components of the steady state vector "converge" to a value, thereby causing them to be proportional to the number of times the surfer visits the n pages (nodes) (Shum).
The number of iterations it takes for the vector to converge and to reach the steady state in this way is called the rate of convergence.

Dangling Node
A directed graph (and, thus, a web graph) can be represented as a matrix, specifically as an adjacency matrix, which can then be transformed into a transition/stochastic matrix. (This construction process is detailed in the section below.) A dangling node is a node that has no out-going edges (but it may have incoming edges). Once a dangling node is reached, no other state can be reached after this, as there are no out-going links. In the matrix representation of such a graph, the dangling node is visualized as a column of zeros. A matrix representation of such a directed graph having a column of zeros in column j means that node j is a dangling node. Such a column of zeros, however, causes to matrix to not be stochastic, as this column would not sum to 1, meaning the PageRank algorithm would not be validily applied on such a matrix, as we'd see the steady state vector converge to zero. The remedy for this is to essentially alter the column of zeros of the matrix representation of a web graph with a dangling node to be a column of $\frac{1}{n}$ as explained in the subsequent section.

Reducible/Irreducible Graph
A graph is irreducible if "for any pair of distinct nodes, we can start from one of them, follow the links in the web graph and arrive at the other node, and vice versa" (Shum). Essentially, this means that any node is reachable from any other node via edges. On the other hand, a reducible graph is the opposite of an irreducible graph: a reducible graph has a set of nodes that have no out-links, such that once a random walker, for instance, arrives at one of these nodes, he/she gets "stuck".

Perron-Frobenius Theorem
Let M be a n x n stochastic, or Markov, matrix with all positive entries.

1. M has a unique steady-state vector, x. If x_0 is any initial state, then $Mk * x_0$ converges to x as $k\infty$. (As stated by Jauregui.)
2. M has only one eigenvalue 1. All other eigenvalues $|\lambda| < 1$
3. The eigenvector corresponding to eigenvalue 1 has all positive entries (As stated by Sternberg.)

Power Method Convergence Theorem

let M be an $n \times n$ stochastic matrix and let $\mathbf{x}$ be its eigenvector corresponding tothe eigenvalue 1. Let $\mathbf{z}$ be the column vector with all entries e(e integer k, $M^k z$ converges to vector $\mathbf{x}$ as $k \to \infty$. (As stated by Tanase and Redu.)

3 Construction/Steady State of Markov Matrix

Here, we talk about key fundamental matrix attributes and graph construction choices that are heavily utilized in the PageRank algorithm, as well as our implementation of the algorithm in 5.

3.1 Constructing Adjacency and Transition Matrix from Directed Graph

Given a list of websites and the websites that they are linked to, we can create a graph by visualizing the websites as nodes and their links as directed edges.

To construct an adjacency matrix A, we first note the number of nodes in the graph. If n is the number of nodes in the graph, then the matrix is $n \times n$. Then, analyzing the graph (and letting i represent the row number and j represent the column number of matrix A) allows us to find the element A_{ij}. Since we're analyzing a directed graph, the edges between nodes have only one direction. If there is an edge going from node j to node i, we say that $A_{ij} = 1$, and otherwise, $A_{ij} = 0$. The adjacency matrix thus allows us to visualize a directed graph in matrix form.

Now, we must construct a transition, or stochastic, matrix from the adjacency matrix in order to describe the transitions according to probabilistic tendencies from one state to another (here, from one node, or page, to another). By definition, we know that this means that the columns of the resulting matrix must sum to 1. Thus, to normalize the adjacency matrix like so we first find the value S(j), which is the sum of the values in the column. Then, assuming that $S(j) \neq 0$, we multiply every column by $\frac{1}{S(j)}$. What $\frac{1}{S(j)}$ represents is an evenly distributed probability in which a node will randomly link to another node.
For example, if there are two directed edges that lead out of a node, assuming that each edge is weighted evenly, there is a $\frac{1}{2}$ chance of a walk randomly selecting either edge to travel to another node. Note that the resulting matrix is a stochastic matrix, as each of the values is non-negative and each column sums to 1. Thus, assuming that no column sums to zero in our adjacency matrix and assuming that the probability of an edge going to a node is equivalent to the probability of this edge going to another node (i.e. the edges are wieghted evenly), we see that our summation and normalization technique above is equiv-

alent to defining our n x n transition matrix as:

$$\begin{cases} A_{ij} = \frac{1}{S(j)} \text{ if node } j \text{ connects to node } i \\ A_{ij} = 0 \text{ otherwise} \end{cases}$$

where D(j) represents the number of outlinks (i.e. outgoing edges) from node j. This is stated by Shum.

3.2 Considering Dangling Nodes

Because we are creating a matrix from a directed graph, it is possible that a node does not have any links leading out of it. In this case, we will have a column entirely of 0s, which will cause an error when we attempt to multiply by $\frac{1}{S(j)}$ as $S(j) = 0$. Thus, our above construction of the transition matrix overlooks the possibility of a dangling node in our directed graph. As previously defined, a dangling node is a node with no outgoing edges. A random walker, upon reaching a dangling node, would reach a "dead end", which, in a web graph, would signify a "page that has no successors has nowhere to send its importance. Eventually, all importance will leak out of the Web" (Stanford). This can formally be seen by seeing that the steady state vector of such a matrix (as defined in 1), which describes the convergence of probabilities, would essentially have all its components converge to zero. Thus, in any case, a dangling node would not accurately represent the page rank of the web pages that link into it, as it would essentially accumulate the probability of the nodes linking into it.
To remedy this, we can replace every 0 in the column of zeros with $\frac{1}{n}$, where n is the number of rows in the matrix A and n is the number of nodes in the directed graph. By doing this, we essentially model the idea that a random walker, after reaching a dangling node, will "jump" to another any other one of the n pages with equal probability. Assigning every zero to $\frac{1}{n}$ in this way allows us to remedy this issue, as now the sum of the columns is not zero but rather 1, just as it should be in a transition matrix depicting probabilities. Formally, then, we can revise our construction of the transition matrix to account for dangling nodes: Bij = 1/P(j) if node j connects to node i 1/n if node j has no outgoing edges 0 otherwise

$$\begin{cases} A_{ij} = \frac{1}{S(j)} \text{ if node } j \text{ connects to node } i \\ A_{ij} = \frac{1}{n} \text{ if node } j \text{ is dangling} \\ A_{ij} = 0 \text{ otherwise} \end{cases}$$

3.3 Considering Damping Factor and Reducible Graphs

Although sometimes not apparent in the adjacency and/or transition matrix representation of a directed graph, our directed graph could be reducible (i.e.

that a set of nodes in the graph have no out-links). The ranking of the collection of pages that have no out-links will become higher than those that do have out-links, as the links will essentially "collect" in the nodes that have no out-links. This phenomenon is similar to the "leak" described in **3.2**. This can formally be seen by how the steady state vector of such a matrix will display the probabilities of the nodes with no outgoing edges as higher than the probabilities of the nodes with outgoing edges. Yet, since these outgoing edges could have incoming edges, it seems as if they should have greater than zero probabilities. Thus, in their creation of the PageRank algorithm, Brin and Page defined the so-called Google matrix M and defined it (as Rousseau explains) like so:

$$M = dA + (1 - d)Q$$

where d is the *damping factor*, matrix A is an nxn stochastic transition matrix of probabilities for the web graph (altered as needed if a dangling node exists), and matrix Q is an nxn matrix such that

$$Q_{ij} = \frac{1}{n} \ \forall (i, j) \text{ such that } 1 < i, j < n.$$

This matrix is essentially a second stochastic matrix to account for the case of a reducible graph. We see that this matrix remedies the situation of the probabilities between nodes with outgoing edges and the group of nodes with no outgoing edges (i.e. the nodes that cause the graph to be reducible) being unequal.

This is because in the random walker interpretation of this matrix M, the damping factor allows for the walker to, with probability d, randomly choose an edge to move to another node from his current node, by essentially "clicking" on a random link on the page. However, to account for reducibility and a group of nodes with no outlinks, with probability $1 - d$, the walker would essentially "surf to a completely random page", which then remedies the reducible graph issue as the algorithm would simply restart its random walk at a random node (Margalit and Rabinoff).

In the PageRank algorithm, the damping factor is normally set to 0.85, just as Page and Brin set it in their Google matrix. This is because, as seen in the above formulation of the matrix, if d = 1, then the issue of the collection of nodes with no outgoing edges absorbing the page rank of the other pages would not be remedied, as the matrix Q would go to zero. On the other hand, if d = 0, then the random walker always restarts randomly (albeit, in a uniformly distributed away). Thus, setting the damping factor between 0 and 1 allows us to find a weighted average between the two edge cases. Specifically, the exact value of the damping factor is dependent on how efficient the PageRank algorithm is; the effect of the specific chosen value of d and the inner workings of this algorithm are further discussed in **7.1**.

3.4 Steady State of Markov Matrix

Ultimately, through the construction of these matrices, the goal is to find a steady state vector of the resulting matrix, which can be seen as the Google matrix. We want to find the steady state vector, because it is this vector that describes the convergence of probabilities and, thus, gives us insight into the ranking of the pages, as it shows where a random walker would most likely end up in a given directed web graph. There are essentially two ways to find the steady state of a Markov Matrix. Firstly, via the Perron Frobenius theorem, we know that the Markov matrix has a unique eigenvector corresponding to the eigenvalue of 1, and this is known as the steady state vector. The second way to find the eigenvector, however, is the reason that this method of simply taking the eigenvector of eigenvalue 1 as the steady state works. This method works because eigenvectors for a Markov matrix M form a basis for R^2 (if there are 2 distinct eigenvectors each with distinct eigenvalues) and, thus, any $v \in \mathrm{R}^2$, can be written as a linear combination of these two vectors. Thus, multiplying M by v and then M^2 by v (and so on), where v is written as a linear combination of the two eigenvectors, shows us that, since the other eigenvalue is less than 1 (for Markov matrices), this term will essentially disappear from our calculations, showing us that this process essentially converges to the vector corresponding to eigenvalue 1 i.e. the steady state vector. This process is outlined in the power method. However, in our algorithm, we simply define the steady state as the eigenvector corresponding to eigenvalue 1.

4 Explanation of PageRank

PageRank is a ranking system originally designed to organize the pages on the web via a ranking system. Under this system, a page has a higher PageRank score if it has more pages linking to it. Furthermore, even if just one important page links to a page, the page being linked to is considered to have a high PageRank score (though it's important to note that any one page's importance is "divided over its out-links" in order to "limit the influence of any one page" (Spielman). This latter piece of information is important, for it ensures that the importance of pages is not just measured by the "naive number of links into the page." This is relevant, for a spammer could create many pages that all mutually link to each other and while the naive measure would label these pages as "important", PageRank would dictate that these pages are in fact not important, for no "outer" important pages link to any of them.

PageRank utilizes the link structure of the web, in which the web is looked at as a web graph, or a directed graph with the nodes representing the pages and the edges represent the links between the pages. Using this graph structure, we see how each web page has in-links, or "links pointing to the page from

other pages" and out-links, or "links found in the page" that are thus directed outwards to other pages. Via analysis of this linking with linear algebra through the construction of web stochastic matrices, we see how PageRank essentially ranks the nodes of a web graph.

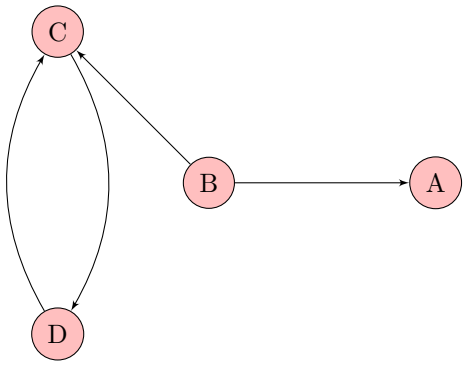
5 PageRank Algorithm

In this section, we detail an example, as well as explain our code implementation. One major choice we made in constructing our algorithm was to use a sort of modular programming approach, meaning we emphasized splitting our tasks into separate functions in order to make our approach more readable.

5.1 Example Along with Code Implementation

5.1.1 Creating A Matrix

We are given the below matrix, in which every node is a website and every directed edge represents a link. Notice how, in this graph, node A is a dangling node, and nodes C and D make this a reducible.



The adjacency matrix below corresponds to this graph:.

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

The Julia Code is shown below:

```
A = [0 1 0 0 ; 0 0 0 0; 0 1 0 1; 0 0 1 0]
```

Clearly, the zeros in column 1 result from the dangling node.

5.1.2 Normalizing, Adjusting for Dangling Nodes

When we normalize a matrix, we essentially turn it into a Markov matrix. Based on our definition in **2**, we know that a Markov Matrix must have columns that sum to 1. In order for this be true, we divide every column by a value described in **3.1** and correct for dangling nodes as described in **3.2**. Our normalized matrix, as well as code for these functions, is shown below.

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

```
using LinearAlgebra

function normalizeAdjacencyMatrix(A)
    n = size(A)[1]  # n = number of rows/cols in A
    # Loop through each value in inputted matrix A.
    for i = 0:(n-1)
        sum = 0
        for j = 1:n
            sum = sum + A[i*n+j]
        end
        # replacing any zero column value (dangling
    node) with 1/n.
        if sum == 0
            for j = 1:n
                A[i*n+j] = 1/n
            end
        # else, dividing by the sum of the column.
        else
            for j = 1:n
                A[i*n+j] = A[i*n+j] / sum
            end
        end
    end
    return A
end
```

5.1.3 Implementing Damping Matrix

As explain in **3.3**, certain matrices may be reducible and would therefore accumulate the probabilities of the stochastic matrix over time. In order to avoid this, we introduce a damping matrix to the normalized matrix, which will bring

randomness into our PageRank matrix and avoid getting "stuck" between certain nodes.
We use the formula **3.3**, $M = dA + (1-d)\frac{1}{n}Q$ where Q is originally an array of 1s and d is the damping factor.
Our new matrix is:

$$M = \begin{bmatrix} 0.25 & 0.4625 & 0.0375 & 0.0375 \\ 0.25 & 0.0375 & 0.0375 & 0.0375 \\ 0.25 & 0.4625 & 0.0375 & 0.8875 \\ 0.25 & 0.0375 & 0.8875 & 0.0375 \end{bmatrix}$$

```
function dampingMatrix(A)
    n = size(A)[1] # number of rows/columns of matrix
    A
    A = normalizeAdjacencyMatrix(A) # normalize matrix
    using 5.1.2

    # Create damping matrix Q
    Q = ones(n,n)
    Q = Q*(1/n)

    # Set damping factor to 0.85
    dampingFactor = 0.85

    # Create Google matrix M, defined like so:
    M = (dampingFactor)*A + (1-dampingFactor)*Q
    return M
end
```

The reason we handle any dangling nodes before we apply the damping factor is because the damping formula will only result in a Markov matrix if A is already Markov as well. If we applied the damping formula without handling dangling nodes, then the 0 column for the dangling node would only sum to d and the matrix M would not be Markov by definition. The importance of a Markov matrix is discussed more in the next section.

5.1.4 Finding Steady State PageRank Vector

Now that we have the dampened Markov Matrix M, we can try to find the steady state vector, which is in this case the PageRank vector. By the Perron-Frobenius Theorem from **2**, we know that there will be a unique eigenvector for this matrix that corresponds to the eigenvalue of 1 (and we also know that this eigenvalue is the greatest out of all). Repeated powers of this matrix will converge to this eigenvector, which can be found easily through the following Julia code.

```
function findSteadyState(A)
    vecA, vecB = eigen(A)  # vecA is eigenvalues, vecB
    is eigenvectors
    steadyStateVector = []
    n = size(A)[1] #row/col size of A
    i = 1
    # Loop through matrix of eigenvectors i.e. vecB.
    # Since eigen returns the eigenvalues in order
   from greatest to least, we retrieve the first
   column i.e. the values of the eigenvector
   corresponding to eigenvalue 1
    while i < length(vecB)
        if (i<= n)
    # Append each value in the first column to list
   steadyStateVector
            append!(steadyStateVector, [vecB[i]])
        end
        i += 1
    end
    # Normalize this vector by dividing each value by
   the sum of the components of the eigenvector
    sumOfComponents = sum(steadyStateVector)
    normalizedSteadyState = (1/sumOfComponents)*
   steadyStateVector
    return normalizedSteadyState
end
```

The reason why we can simply find the steady state eigenvector instead of repeatedly multiplying the Markov matrix is further discussed in **3.4**.
The PageRank vector is:

$$p = \begin{bmatrix} 0.077 \\ 0.054 \\ 0.441 \\ 0.429 \end{bmatrix}$$

This vector tells us that in a random walk's steady state, the walk will be on node C, the third node, for approximately 44% of the time. These steady state probabilities tell us the ranks for every node in the part of a graph.

6 HITS Algorithm

The HITS (Hyperlink-Induced Topic Search) Algorithm was developed by Jon Kleinberg of Cornell University around the same time as the PageRank. The

HITS algorithm is used largely because of its more intelligent form of searching, which relies on more than just links and text (Tanase and Redu). The two major players in the HITS algorithm are authoritative pages and hubs.
Authoritative pages are websites that contain valuable information on the subject of a given query. Hubs are pages that are link to many authoritative pages. Because of this, authoritative pages and hubs are mutually connected: a page is more authoritative if it links to many hubs, and a page is a better hub if it links to many authoritative pages.
The authority vectors and hub vectors are detailed below (University of Cincinnati):

$$\begin{aligned} \mathbf{a} &= A^T h \qquad \mathbf{h} = Aa \\ \Rightarrow \mathbf{a} &= A^T Aa \qquad \mathbf{h} = AA^T h \end{aligned}$$

These equations make it clear how **a** and **h** are dependent on eachother. These equations also make it apparent that **a** and **h** are eigenvectors of the matrices $A^T AandAA^T$, respectively. This allows us to find the authoritative and hub vectors. The following Julia code performs this function, on the same adjacency matrix as in **5.1**.

```
function hits(A)
    #creating the matrices
    authMatrix = transpose(A)*A
    hubMatrix = A*transpose(A)
    #finding the eigenvector corresponding to the
    #highest eigenvalue
    #the largest eigenvalue won't necessarily be 1
    #however, a slightly modified findSteady will
   first search for
    #the largest eigenvalue, and then find the
   corresponding vector
    a = findSteady(authMatrix)
    h = findSteady(hubMatrix)
    return a, h
end
hits(A)
```

7 Final Thoughts

7.1 Efficiency

Because the PageRank algorithm is implemented in systems with billions of nodes and edges, for very large graphs, and the efficiency of PageRank is measured by its rate of convergence. This rate of convergence can be changed based on the damping factor d. The smaller d is, the faster the rate of convergence, as less randomness is artificially brought upon the matrix. In fact, the number of iterations required to achieve the rate of convergence is approximately d (Langville and Meyer). However, having a smaller d may result in certain nodes being unfairly favored. For example, there may be many "spam" websites leading to a single website, or one very powerful website creating too much "pull" towards its direction. In a compromise between rate of convergence efficiency and algorithmic truthfulness, Google chose a damping factor of 0.85. Furthermore, another aspect of efficiency could be related to how the number of iterations required to achieve the steady state, which is what gives us the ranking of the pages, could be cut down by normalizing the resulting vector every time it is multiplied by the Markov matrix.

7.2 Successors and Issues

Though the degree of randomness introduced by the damping factor increases the fairness of ranking algorithms, there are still a couple open issues present. In fact, one of the main disadvantages of PageRank is that it does not take the content of a site into account (Devi, Gupta, and Dixit). It is entirely possible that a website is ranked very highly because of related links, despite not catering to the subject of the query (Tanase and Redu). Many successor algorithms, such as HITS, attempt to remedy this by taking the content of a website into account. This allows for more content-driven rankings, and perhaps more relevant results. However, Google's PageRank algorithm maintains relative fairness as well due to its carefully chosen damping factor, and PageRank's superior efficiency compared to algorithms such as hits makes it a more viable solution for modern-day search engines that handle millions of queries

8 Conclusion

The PageRank algorithm, as well as its application of Markov chains, has arguably had one of the largest impacts on society in the past couple of decades. By applying and innovating upon Linear Algebra concepts, PageRank revolutionized the internet and changed the way in which the average person interacted with their world.

9 References

Schmidt, Andrea (n.d.). Retrieved from https://www.mit.edu/ kardar/teaching/projects/chemotaxis(AndreaSchmidt)/random.htm. Accessed 6 December 2019.

Broadley, C. (2019, November 6). AltaVista – The Tragic Tale Of The Search Enigine Pioneer. Retrieved from https://digital.com/about/altavista/. Accessed 6 December 2019.

Demystifying the PageRank and HITS Algorithms. (2013, February 5). Retrieved from https://cs7083.wordpress.com/2013/01/31/demystifying-the-pagerank-and-hits-algorithms/ Accessed 6 December 2019.

Devi, P., Gupta, A., Dixit, A. (2014). Comparative Study of HITS and PageRank Link based Ranking Algorithms. International Journal of Advanced Research in Computer and Communication Engineering. Accessed 6 December 2019.

Shum, Kenneth. ENGG2012B Advanced Engineering Mathematics Notes on PageRank Algorithm. Retrieved from http://home.ie.cuhk.edu.hk/ wkshum/papers/pagerank.pdf.Accessed 6 December 2019.

Jauregui, J. (n.d.). Math 312 Markov chains, Google's PageRank algorithm. Retrieved from https://www.math.upenn.edu/ kazdan/312F12/JJ/MarkovChains/ markov google.pdf.Accessed 6 December 2019.

Knill, O. (n.d.). Lecture 34: Perron Frobenius theorem. Accessed 6 December 2019.

Knowledge Base: Internal Linking And Pagerank Indicators. (n.d.). Retrieved from https://www.botify.com/support/internal-pagerank-indicators. Accessed 6 December 2019.

Langville, A., Meyer, C. (n.d.). Deeper Inside PageRank. Internet Mathematics Vol. 1, No. 3: , 335–380. Retrieved from https://galton.uchicago.edu/ lekheng/meetings/mathofranking/ref/langville.pdf. Accessed 6 December 2019.

Margalit, D., Rabinoff, J. (n.d.). Interactive Linear Algebra. Retrieved from https://textbooks.math.gatech.edu/ila/stochastic-matrices.html. Accessed 6 December 2019.

Markov Chains. (n.d.). Retrieved from https://brilliant.org/wiki/markov-chains/.Accessed 6 December 2019.

Sternberg, S. (n.d.). Lecture12: The Perron-Frobenius theorem. Retrieved from http://www.math.harvard.edu/library/sternberg/slides/1180912pf.pdf. Accessed 6 December 2019.

Tanase, R., Radu, R. (n.d.). Lecture 3. Retrieved from http://pi.math.cornell.edu/ mec/Winter2009/RalucaRemus/Lecture3/lecture3.html. Accessed 6 December 2019.

Lecture 4: HITS Algorithm - Hubs and Authorities on the Internet. (n.d.). Retrieved from http://pi.math.cornell.edu/ mec/Winter2009/RalucaRemus/Lecture4/lecture4.html. Accessed 6 December 2019.

Rousseau, Christiane. (2015, May 30). Klein Project. How Google works: Markov chains and eigenvalues. Retrieved from http://blog.kleinproject.org/?p=280. Accessed 6 December 2019

Spielman, D. (2010, October 28). PageRank and Random Walks on Directed Graphs. Retrieved from http://www.cs.yale.edu/homes/spielman/462/2010/lect16-10.pdf. Accessed 6 December 2019