LINEAR ALGEBRA PROJECT SPRING 2023

TEAM 6

Exploring the Linear Algebra behind the milestone of the 90s:
The Google PageRank Algorithm

FINAL REPORT

1 Introduction

Google PageRank is a link analysis algorithm developed by Larry Page and Sergey Brin, the founders of Google, that revolutionized web search and web indexing and became a cornerstone of the company's success.

PageRank's objective is to quantify the relative authority and **relevance** of web pages so that search engines can **rank and retrieve** search results in a way that meets users' information demands. PageRank utilizes a sophisticated mathematical framework that uses concepts from linear algebra, particularly **eigenvalues and eigenvectors**, that calculate the importance scores of web pages.

In this project, we explore the mathematics and algorithms that underlie PageRank, looking at how principles from graph theory and linear algebra are used to rate the significance of web sites.

2 State of the Art Literature

- "The Anatomy of a Large-Scale Hypertextual Web Search Engine" by Sergey Brin and Lawrence Page (1998) [1]

  This groundbreaking article presented the PageRank algorithm and gave a thorough rundown of Google's web search engine. It discussed the significance of **link analysis** and how PageRank scores are used to rank web pages.

- “The PageRank Citation Ranking: Bringing Order to the Web” by Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd (1999) [2]

  This paper provided an in-depth analysis of the PageRank algorithm and its application in web search. It discussed ”,the relationship between PageRank and the citation structure of academic papers, demonstrating how the algorithm can be used to rank scholarly articles.

- “MATHEMATICS BEHIND GOOGLE’S PAGERANK ALGORITHM”,by BRIAN MOOR

  The thesis explores the history and mathematics behind the PageRank algorithm, and the optimizations and expanded uses it has found over the years.The paper takes a deep look into the PageRank algorithm, the methods it contains, and the other places where it can be used. It examines the background mathematics necessary for the understanding of the algorithm, and walks through an example system building up to the PageRank algorithm.

- “ The $25,000,000,000* Eigenvector: The Linear Algebra Behind Google” Kurt Bryan and Tanya Leise

  It provides an in-depth analysis of the mathematics and linear algebra behind the PageRank algorithm used by Google. This paper was published in the Society for Industrial and Applied Mathematics (SIAM) Review in 2006.

  - The paper starts by introducing the significance of PageRank in web search and its impact on the ranking of web pages.
  - It highlights the importance of linear algebra, specifically eigenvectors and eigenvalues, in understanding the core principles of PageRank.

  Background and Mathematical Formulation

  - The authors explain the fundamental concepts of Markov chains, transition matrices, and random walks on graphs.
  - They present the basic mathematical formulation of PageRank as a steady-state probability distribution over web pages based on the random surfer model.

Power Iteration Method

- The paper discusses the power iteration method as an iterative algorithm for computing the dominant eigenvector of the PageRank matrix.
- It explains the steps involved in the power iteration method and its convergence properties.

Matrix Notation and Implementation

- The authors introduce the matrix notation used in PageRank calculations and describe the implementation details of the algorithm.
- They discuss the challenges of handling large-scale web graphs and strategies for efficient computation.

Additional Considerations

- The paper covers additional considerations in PageRank, including the damping factor, teleportation vector, and handling of sink nodes (pages with no outgoing links).
- It explains the impact of these factors on the stability and convergence of the algorithm.

Numerical Examples and Results

- The authors provide numerical examples to illustrate the calculations involved in computing PageRank scores.
- They discuss the influence of different parameters on the ranking results and provide insights into the behavior of the algorithm.

Extensions and Generalizations

- The paper briefly mentions extensions and variations of PageRank, such as personalized PageRank and topic-sensitive PageRank.
- It highlights potential research directions and applications related to the PageRank algorithm.

3 Problem Statement

Our project aims to answer the following queries/problems:

- How does the PageRank algorithm use linear algebra to determine a web page's relevance score?
- What connection exists between the web graph's link structure and the relevance ratings given by PageRank?

- How does the iterative PageRank calculation technique help the significance scores to converge and remain stable?
- What are the implications of the damping factor in controlling the flow of importance and ensuring the effectiveness of PageRank?

4 Use Case of Linear Algebra in the Project

- Transition Matrix: The transition matrix is a square matrix where each element represents the probability of transitioning from one page to another in a web graph. The transition matrix can be efficiently handled and analyzed using linear algebra techniques.
- Eigenvector Calculation: The PageRank algorithm utilizes eigenvalues and eigenvectors to calculate the importance scores of web pages.

  The transition matrix is analyzed to find the dominant eigenvector, which represents the relative importance of each page. Linear algebra techniques like matrix-vector multiplication are used to redistribute importance across connected pages and cause the scores to converge to a stable state.
- Convergence and Stability: Linear algebra helps ensure the convergence and stability of PageRank scores. PageRank can analyze the flow of importance through the web graph using eigenvalues and eigenvectors, and it can then be constrained using a damping factor.

5 BUILDUP OF GOOGLE PAGERANK

The PageRank(as the name suggests, ranking of pages) algorithm, which rates the significance of webpages based on an eigenvector of a weighted link matrix, is largely responsible for Google's success. Analysis of the PageRank formula provides a wonderful applied topic for a linear algebra course.

In the late 1990s, search engines other than Google had to wade through screen after screen of links to irrelevant web pages that just happened to match the search text. The Google PageRank was a milestone at that time as it quantitatively rated the importance of each page on web, allowing Google to rank the pages and thereby present to the user the more relevant and important pages first. This ranking of pages is termed as "Google PageRank".

Anyone creating a website they want people to visit frequently has to understand how to calculate PageRank because getting listed first in a Google search results in lots of people viewing your page.

The amount of information on the web is growing rapidly. **Automated search engines that rely on keyword matching usually return too many low quality matches. To make matters worse, some advertisers attempt to gain people's attention by taking measures meant to mislead automated search engines.**

Google PageRank makes especially heavy use of the additional structure present in hypertext to provide much higher quality search results.

"We chose our system name, Google, because it is a common spelling of googol, or 10^{100} *and fits well with our goal of building very large-scale search engines."*

- Sergey Brin and Larry Page

The goal of the PageRank system is to address many of the problems, both in quality and scalability, introduced by scaling search engine technology to such extraordinary numbers. Search engines such as Google have to do three basic things:

- Crawl the web and locate all web pages with public access.
- Index the data from step 1, so that it can be searched efficiently for relevant keywords or phrases.
- Rate the importance of each page in the database, so that when a user does a search and the subset of pages in the database with the desired

information has been found, the more important pages can be presented first.

This project will focus on step 3 (the ranking part).

5.1 Background Mathematics

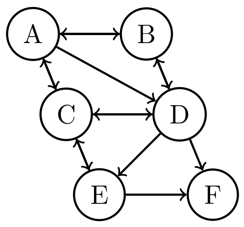
5.1.1 GRAPH THEORY

One of the essential foundations necessary for representing the Internet network is graph theory. A graph is made up of two sets, a set of vertices, V , and a set of pairs of vertices, the edges, E.

The undirected graph in Figure below can be written as:

V (G) = {A, B, C, D, E, F}

E(G) ={(A, B),(A, C),(A, D),(B, A),(B, D),(C, A),(C, D), (C, E),(D, B),(D, C),(D, E),(E, C),(E, F),(F, D)}



Directed Graph **1.1**

In the above figure, A,B,C,D,E,F can be **WEB PAGES** on Internet. **The → in graph from X to Y indicate webpage X contains link which links it to webpage Y.**

$$\begin{bmatrix} & A & B & C & D & E & F \\ A & 0 & 1 & 1 & 0 & 0 & 0 \\ B & 1 & 0 & 0 & 1 & 0 & 0 \\ C & 1 & 0 & 0 & 1 & 1 & 0 \\ D & 1 & 1 & 1 & 0 & 0 & 1 \\ E & 0 & 0 & 1 & 1 & 0 & 0 \\ F & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Adjacency Matrix Representation of Graph

Vertex	Adjacent Vertices
A	B, C, D
B	A, D
C	A, D, E
D	B, C, E
E	C, F
F	D

Adjacency List Representation of Graph

5.1.2 NOTATION

The notation and mathematics we will use in this paper are explained here. All matrices are n × n, and vectors are n × 1, and all matrix entries are real numbers. The transpose of a vector $\boldsymbol{v}$ is the 1 × n vector $\boldsymbol{v}^T$. The vector **e** is a column vector of all ones. PR(A) is the PageRank of the page A, the damping factor is d, conventionally defined by **Brin and Page** (1998) as 0.85, and the number of pages linked is N.

5.1.3 APPLICATION TO THE INTERNET

The Internet is made up of many webpages, each containing information on its page. Each page may also contain links to other pages. This can be modeled using graph theory, where each page is a vertex, and the edges between each vertex are equivalent to the links from one page to another.

The model shown in "Directed Graph 1.1" is a simplified set of webpages showing a graph of directed edges, representing pages and the outgoing links between them. This simplified model of the Internet is easily expandable and is useful for describing the methods used in the PageRank algorithm.

5.1.4 STOCHASTIC MATRIX

A stochastic matrix, also known as a probability matrix or a Markov matrix, or a column stochastic matrix is matrix with the sum of elements in each column equal to one, used in the study of Markov chains and probability theory. In a stochastic matrix, its eigenvalues are guaranteed to have a maximum magnitude of 1, or have a complex number as an eigenvalue with a magnitude less than or equal to 1. The eigenvector corresponding to the eigenvalue 1 is of particular interest as it represents the stationary distribution or equilibrium distribution of a Markov chain associated with the stochastic matrix.

Given A is a stochastic matrix, so $\lambda = 1 : |A - \lambda I| = 0$, $A\boldsymbol{v} = \lambda\boldsymbol{v}$

As $\lambda = 1$, $A\boldsymbol{v} = \boldsymbol{v}$

where A is a transition matrix and $\boldsymbol{v}$ is a probability vector. As an Example:

$$\begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{3} & 0 & 0 & 0 \\ \frac{1}{3} & 0 & 0 & \frac{1}{4} & 0 & 0 \\ \frac{1}{3} & 0 & 0 & \frac{1}{4} & \frac{1}{2} & 0 \\ \frac{1}{3} & \frac{1}{2} & \frac{1}{3} & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{3} & \frac{1}{4} & 0 & 1 \\ 0 & 0 & 0 & \frac{1}{4} & \frac{1}{2} & 0 \end{bmatrix}$$

COLUMN STOCHASTIC MATRIX

5.1.5 MARKOV CHAINS

Transition Matrix

The transition matrix, denoted as P, is an $n \times n$ matrix where P_{ij} represents the probability of transitioning from state s_j to state s_i. The elements of each column of P must sum to 1.

$$P = \begin{bmatrix} P_{11} & P_{12} & \cdots & P_{1n} \\ P_{21} & P_{22} & \cdots & P_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ P_{n1} & P_{n2} & \cdots & P_{nn} \end{bmatrix}$$

Probability Vector

The probability vector, denoted as p, represents the probabilities of being in each state at a specific time step. It is typically a column vector of probabilities.

$$\boldsymbol{p} = \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{bmatrix}$$

Long-term Iteration

To compute the long-term behavior of the Markov chain, we iterate the transition matrix P and a probability vector $\boldsymbol{x}$ over multiple time steps until convergence to the stationary distribution. This process can be represented as:

$$\boldsymbol{x}_t = P\boldsymbol{x}_{t-1}$$

where we hope to arrive at the limit as $t \to \infty$.

5.1.6 Eigenvector and Eigenvalue Calculation

Let A be an $n \times n$ matrix. We want to find the dominant eigenvector x and the corresponding eigenvalue λ of the matrix A.

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$$

To calculate the eigenvector x, we solve the equation $Ax = \lambda\boldsymbol{x}$ as follows:

$$|A - \lambda I| = 0$$

This holds since

$$\begin{bmatrix} a_{11} - \lambda & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} - \lambda & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} - \lambda \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

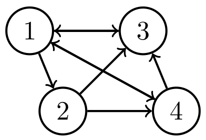
Given A is a stochastic matrix, so $\lambda = 1$:
$|A - \lambda I| = 0$, $Ax = \lambda\boldsymbol{x}$

As $\lambda = 1$, $Ax = \boldsymbol{x}$ where A is a stochastic transition matrix and $\boldsymbol{x}$ is a probability vector.

5.2 Developing a formula to rank pages

1. **Basic Idea**

The importance score for any web page will always be a non-negative real number. A core idea in assigning a score to any given web page is that the page's score is derived from the links made to that page from other web pages. The links to a given page are called the backlinks for that page. The web thus becomes a democracy where pages vote for the importance of other pages by linking to them.



An example of a web with only four pages. An arrow from page A to page B indicates a link from page A to page B.
Figure 2.1

$$\left[\begin{array}{ccccc} & 1 & 2 & 3 & 4 \\ 1 & 0 & 0 & 1 & 1 \\ 2 & 1 & 0 & 0 & 0 \\ 3 & 1 & 1 & 0 & 1 \\ 4 & 1 & 1 & 0 & 0 \end{array}\right]$$

Suppose the web of interest contains n pages, each page indexed by an integer k, $1 \leq k \leq n$. A typical example is illustrated in Figure above, in which an arrow from page A to page B indicates a link from page A to page B. We'll use x_k to denote the importance score of page k in the web. The x_k is non-negative, and $x_j > x_k$ indicates that page j is more important than page k (so $x_j = 0$ indicates page j has the least possible importance score).

SIMPLE SOLUTION AND ITS DISADVANTAGE

A very simple approach is to take x_k as the number of backlinks for page k. In the example in Figure 2.1, we have $x_1 = 2$, $x_2 = 1$, $x_3 = 3$, and $x_4 = 2$, so that page 3 would be the most important, pages 1 and 4 tie for second, and page 2 is least important. A link to page k becomes a vote for page k's importance.

Disadvantage

This approach ignores an important feature one would expect a ranking algorithm to have, namely, that a link to page k from an important page should boost page k's importance score more than a link from an unimportant page.

For example, **a link from YAHOO ought to boost your page's score much more than a link from a local webpage**

In the web of Figure above, pages 1 and 4 both have two backlinks: each links to the other, but page 1's second backlink is from the seemingly important page 3, while page 4's second backlink is from the relatively unimportant page 1. As such, perhaps we should rate page 1's importance higher than that of page 4.

ALTERNATIVE SOLUTION

As a first attempt at incorporating this idea, let's compute the score of page j as the sum of the scores of all pages linking to page j. For example, consider the web of Figure 2.1. The score of page 1 would be determined by the relation $x_1 = x_3 + x_4$. Since x_3 and x_4 will depend on x_1, this scheme seems strangely self-referential, but it is the approach we will use, with one more modification. Just as in elections, we don't want a single individual to gain influence merely by casting multiple votes.

In the same vein, we seek a scheme in which a web page doesn't gain extra influence simply by linking to lots of other pages. If page j contains n_j links, one of which links to page k, then we will boost page k's score by $\frac{x_j}{n_j}$, rather than by x_j. In this scheme, each web page gets a total of one vote, weighted by that web page's score, that is evenly divided up among all of its outgoing links. To quantify this for a web of n pages, let $L_k \subset \{1, 2, ..., n\}$ denote the set of pages with a link to page k, that is, L_k is the set of page k's backlinks.

For each k we require

$$x_k = \sum_{j \in L_k} \frac{x_j}{n_j}$$

where n_j is the number of outgoing links from page j (which must be positive since if $j \in L_k$ then page j links to at least page k!).

Let's apply this approach to the four-page web of Figure above 2.1. For page 1, we have $x_1 = \frac{x_3}{1} + \frac{x_4}{2}$, since pages 3 and 4 are backlinks for page 1 and page 3 contains only one link, while page 4 contains two links (splitting its vote in half). Similarly, $x_2 = \frac{x_1}{3}$, $x_3 = \frac{x_1}{3} + \frac{x_2}{2} + \frac{x_4}{2}$, and $x_4 = \frac{x_1}{3} + \frac{x_2}{2}$. These linear equations can be written as $Ax = x$, where $x = [x_1\, x_2\, x_3\, x_4]^T$ and A is the coefficient matrix.

$$A = \begin{bmatrix} & 1 & 2 & 3 & 4 \\ 1 & 0 & 0 & 1 & \frac{1}{2} \\ 2 & \frac{1}{3} & 0 & 0 & 0 \\ 3 & \frac{1}{3} & \frac{1}{2} & 0 & \frac{1}{2} \\ 4 & \frac{1}{3} & \frac{1}{2} & 0 & 0 \end{bmatrix}$$

LINK MATRIX

The eigenvalues and eigenvectors x of a matrix A satisfy the equation Ax = x, x != 0 by definition.We thus seek an eigenvector x with eigenvalue 1 for the matrix A. We will refer to A as the "LINK MATRIX" for the given web.

$\rightarrow$ Scale these "importance score eigenvectors" so that the components sum to 1.

In this case we obtain $x_1 = \frac{12}{31} \approx 0.387$, $x_2 = \frac{4}{31} \approx 0.129$, $x_3 = \frac{9}{31} \approx 0.290$, and $x_4 = \frac{6}{31} \approx 0.194$.

Note that this ranking differs from that generated by simply counting backlinks. It might seem surprising that page 3, linked to by all other pages, is not the most important. To understand this, note that page 3 links only to page 1 and so casts its entire vote for page 1. This, with the vote of page 2, results in page 1 getting the highest importance score.

The matrix A for any web must have 1 as an eigenvalue if the web has no **dangling nodes** (pages with no outgoing links). To see this, first note that for a general web of n pages gives rise to a matrix A with $A_{ij} = \frac{1}{n_j}$ if page j links to page i, and $A_{ij} = 0$ otherwise. The j-th column of A then contains n_j non-zero entries, each equal to $\frac{1}{n_j}$, and the column thus sums to 1. This motivates the following definition, used in the study of Markov chains.

The matrix A for a web with no dangling nodes is column-stochastic.

Every column-stochastic matrix has 1 as an eigenvalue.

Proof. Let A be an $n \times n$ column-stochastic matrix, and let e denote an n-dimensional column vector with all entries equal to 1. Recall that A and its transpose A^T have the same eigenvalues. Since A is column-stochastic, it is easy to see that $A^T e = e$, so that 1 is an eigenvalue for A^T and hence for A.

In what follows, we use $V_1(A)$ to denote the eigenspace for eigenvalue 1 of a column-stochastic matrix A.

Unfortunately, it is not always true that the link matrix A will yield a unique ranking for all webs. Consider the web in Figure, for which the link matrix is

$$\begin{bmatrix} & 1 & 2 & 3 & 4 & 5 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 1 & 1 \\ 4 & 1 & 1 & 0 & 0 & 1 \\ 5 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

We find here that $V_1(A)$ is two-dimensional; one possible pair of basis vectors is $x = \left[\frac{1}{2}, \frac{1}{2}, 0, 0, 0\right]^T$ and $y = \left[0, 0, \frac{1}{2}, \frac{1}{2}, 0\right]^T$. But note that any linear combination of these two vectors yields another vector in $V_1(A)$, e.g., $\frac{3}{4}x + \frac{1}{4}y = \left[\frac{3}{8}, \frac{3}{8}, \frac{1}{8}, \frac{1}{8}, 0\right]^T$. It is not clear which, if any, of these eigenvectors we should use for the rankings!

It is no coincidence that for the web of Figure 2.2 we find that $\dim(V_1(A)) > 1$. It is a consequence of the fact that if a web W, considered as an undirected graph (ignoring which direction each arrow points), consists of r disconnected subwebs $W_1, \ldots, W_r$, then $\dim(V_1(A)) \geq r$, and hence there is no unique importance score vector $x \in V_1(A)$ with $\sum_i x_i = 1$. This makes intuitive sense: if a web W consists of r disconnected subwebs $W_1, \ldots, W_r$ then one would expect difficulty in finding a common reference frame for comparing the scores of pages in one subweb with those in another subweb.

Indeed, it is not hard to see why a web W consisting of r disconnected subwebs forces $\dim(V_1(A)) \geq r$. Suppose a web W has n pages and r component subwebs $W_1, \ldots, W_r$. Let n_i denote the number of pages in W_i. Index the pages in W_1 with indices 1 through n_1, the pages in W_2 with indices $n_1 + 1$ through $n_1 + n_2$, the pages in W_3 with indices $n_1 + n_2 + 1$ through $n_1 + n_2 + n_3$, etc. In general, let $N_i = \sum_{j=1}^{i} n_j$ for $i \geq 1$, with $N_0 = 0$, so W_i contains pages $N_{i-1} + 1$ through N_i. For example, in the web of Figure 2 we can take $N_1 = 2$ and $N_2 = 5$, so W_1 contains pages 1 and 2, W_2 contains pages 3, 4, and 5.

6 The Actual PageRank Algorithm

6.1 Definitions

The following is the definition of the PageRank of a webpage, as given by Brin and Page in their original 1998 paper.

$$PR(a_i) := (1-d) + d \cdot \sum_{a_j \in G(a_i)} \frac{PR(a_j)}{L(a_j)} \tag{1}$$

$PR(a_i)$ is the PageRank of a webpage a_i, d is a dampening factor, which can take a value between 0 and 1 (set to 0.85 by them in their paper), and a_j is a webpage which has an outgoing link to a_i. $L(a_j)$ represents the total number of outgoing links in the page a_j.

However, the PageRanks of all webpages do not form a probability distribuition (they don't add up to 1) with this definition, and the $(1-d)$ term must be divided by the total number of webpages, N, to achieve this.

$$PR(a_i) := \frac{(1-d)}{N} + d \cdot \sum_{a_j \in G(a_i)} \frac{PR(a_j)}{L(a_j)} \tag{2}$$

The link matrix, which acts as the transition matrix, is defined as

$$H_{ij} := \begin{cases} 1/L(a_j) & \text{if } a_j \text{ links to } a_i \\ 0 & \text{otherwise} \end{cases}$$

6.2 Damping Factor

The damping factor, denoted by d, is another important component of the PageRank algorithm. It represents the probability that a user will continue following the links on the current page, rather than randomly jumping.

6.2.1 Effect on PageRank Calculation

The damping factor affects the calculation of the PageRank score for each page. When a user follows a link on a page, they have two options: either follow the link on the new page or randomly jump. The damping factor determines the probability of each option.

A higher damping factor assigns more weight to following links, meaning that the user is more likely to continue exploring the web by following the links on the pages they visit. On the other hand, a lower damping factor assigns more weight to random jumping, indicating that the user is more likely to randomly jump to different pages.

6.2.2 Choosing an Optimal Damping Factor

The choice of an optimal damping factor depends on the characteristics of the web graph and the behavior of users. Google typically uses a damping factor of 0.85, which has been found to yield good results in practice. However, in certain cases, different values may be more suitable.

6.3 Iterative Method to calculate PageRank scores

To calculate the final PageRank scores of the webpages, we find the principal eigenvector of the link matrix of the web corresponding to the initial (probability) PageRank vector. To do this, we iteratively multiply the link matrix H with the PageRank vector from the previous iteration. We desire to arrive at the *steady state* PageRank vector, i.e. to compute

$$\lim_{n\to\infty} H^n \boldsymbol{p_0},$$

where $\boldsymbol{p_0}$ is the initial PageRank vector with equal ranks to all the webpages. In practice, we iterate till we obtain only a marginal difference, say ϵ, between two successive iterations. Also, to account for the damping factor, in each iteration, we multiply the matrix-vector product with it, and add another factor that accounts for random jumps to any page in the web the user may take if they are bored of surfing. Thus the recurrence relation becomes

$$\boldsymbol{p_{k+1}} = dH\boldsymbol{p_k} + \frac{(1-d)}{N}\boldsymbol{u} \qquad \forall k > 1 \tag{3}$$

where $\boldsymbol{p_{k+1}}$ is the PageRank vector in the next iteration, $\boldsymbol{p_k}$ is the PageRank vector in the current iteration, and d is the damping factor. $\boldsymbol{u}$ is a column vector with all entries as 1.

Convergence and Stability: The fact that after a number of iterations, the difference between the entries of successive PageRank vectors becomes negligibly small, corresponds to the idea that a sequence of vectors, defined by a matrix transformation applied on the previous iteration to get the new iteration, converges to a final *steady state* vector, which is the principal eigenvector of the matrix. This convergence ensures that the PageRank scores reflect the global importance of a page in the whole web graph, removing bias, instability and noise. Stable scores allow for fair comparison between webpages, providing a reliable means to rank them.

6.4 Example

As an example, consider the directed graph below in Figure 2.

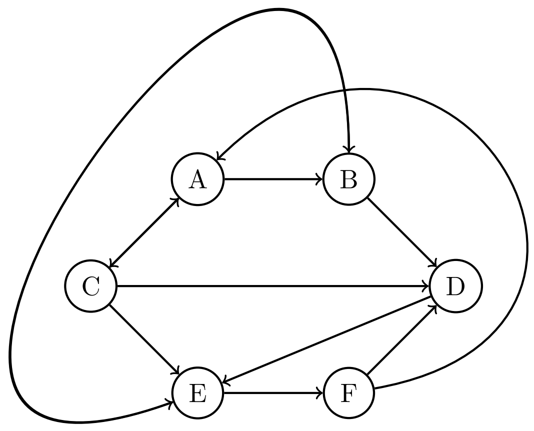


Figure 2

The link matrix for this graph is given by

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

The initial PageRank vector is given by

$$\boldsymbol{p_0} = \begin{bmatrix} 1/6 \\ 1/6 \\ 1/6 \\ 1/6 \\ 1/6 \\ 1/6 \end{bmatrix}$$

The steady state PageRank vector (corresponding to the principal eigenvector of H), computed after 21 iterations, taking into account the damping factor ($d = 0.85$), is given by

$$\boldsymbol{p} \approx \begin{bmatrix} 0.107942 \\ 0.193783 \\ 0.070875 \\ 0.190299 \\ 0.289194 \\ 0.147907 \end{bmatrix}$$

which implies that the ranking of the pages is E, B, D, F, A, C. As predicted the vector's entries sum up to 1.

6.5 Dangling Nodes

So far the example has shown every webpage having an outgoing link. However, it is possible that there are webpages that do not link to any other page. Nodes representing such pages are called "Dangling" nodes. According to the original definition of the link matrix, if a_i is a dangling node, then the i^{th} column would be entirely zero, thus disobeying the stochastic matrix property. And if we compute the final PageRank now, the steady state vector's entries would no longer add up to one.

As an example, suppose we drop off the edge from D to E in the graph given in figure 2.

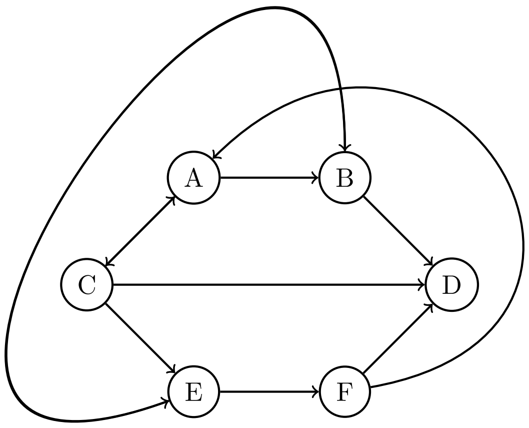


Figure 3

The link matrix for this graph is given by

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

As the 4^{th} column shows, this matrix is not a valid transition matrix. The initial PageRank vector is given by

$$\boldsymbol{p_0} = \begin{bmatrix} 1/6 \\ 1/6 \\ 1/6 \\ 1/6 \\ 1/6 \\ 1/6 \end{bmatrix}$$

If we perform one iteration of the recurrence relation given in equation 3, taking the vector p_0 as the initial PageRank vector, we get

$$\boldsymbol{p} \approx \begin{bmatrix} 0.143056 \\ 0.166667 \\ 0.095833 \\ 0.213889 \\ 0.143056 \\ 0.095833 \end{bmatrix}$$

the sum of whose entries is 0.858334, which shows that the process broke stochasticity of the PageRank vector.

There are several ways to circumvent this, one being removing such nodes entirely, as done by Brin and Page in a follow-up paper$^{[}$*cite*], and another being setting each entry in the i^{th} column to $1/N$ for every dangling node a_i, as the user can be thought to have equal probability to move to any other page in the web, after they end up at a dangling node.

If we take take $\boldsymbol{d}$ as a row vector that identifies the dangling nodes, defined by

$$d_i := \begin{cases} 1 & \text{if } a_i \text{ is a dangling node} \\ 0 & \text{otherwise} \end{cases}$$

and take $\boldsymbol{w}$ as a uniform column vector (each entry $1/N$, then we can define the new (stochastic) link matrix, say M, in terms of the original link matrix H.

$$M = H + \boldsymbol{wd} \tag{4}$$

This new matrix restores the stochasticity of the probability vector containing the PageRank scores.

Continuing the above graph, the corrected link matrix for this graph is given by

$$M = \begin{bmatrix} 0 & 0 & 1/3 & 1/6 & 0 & 1/2 \\ 1/2 & 0 & 0 & 1/6 & 1/2 & 0 \\ 1/2 & 0 & 0 & 1/6 & 0 & 0 \\ 0 & 1/2 & 1/3 & 1/6 & 0 & 1/2 \\ 0 & 1/2 & 1/3 & 1/6 & 0 & 0 \\ 0 & 0 & 0 & 1/6 & 1/2 & 0 \end{bmatrix}$$

On iterating 21 times to find the eigenvector (with damping corrections), we get

$$\boldsymbol{p} \approx \begin{bmatrix} 0.147843 \\ 0.194680 \\ 0.120498 \\ 0.230583 \\ 0.174547 \\ 0.131847 \end{bmatrix}$$

the sum of whose entries is 1, hence it is still a probability vector. Now the rankings are D, B, E, A, F, C.

7 Sample Python code for PageRank Stimulation

```python
import numpy as np
import networkx as nx
import matplotlib.pyplot as plt


def personalized_pagerank(adjacency_matrix, personalization_vector, damping_factor=0.85, max_iterations=100, epsilon=1e-6):
    normalized_matrix = adjacency_matrix / np.sum(adjacency_matrix, axis=0)
    num_nodes = len(adjacency_matrix)
    pagerank_vector = np.ones(num_nodes) / num_nodes
    for _ in range(max_iterations):
        new_pagerank_vector = (1 - damping_factor) / num_nodes + damping_factor * np.dot(normalized_matrix, pagerank_vector)
        if np.linalg.norm(new_pagerank_vector - pagerank_vector) < epsilon:
            break
        pagerank_vector = new_pagerank_vector
    personalized_pagerank_vector = np.multiply(pagerank_vector, personalization_vector)
    personalized_pagerank_vector /= np.sum(personalized_pagerank_vector)
    return personalized_pagerank_vector



def check_and_modify_stochastic(matrix):
    if np.any(np.all(matrix == 0, axis=0)):
        n = matrix.shape[1]  # Number of columns
        col_index = np.where(np.all(matrix == 0, axis=0))[0][0]
        correction = []
        for i in range(0, n):
            correction.append([]);
            for j in range(0, n):
                correction[i].append(0);
        for i in range(0, n):
            if(i!=col_index):
                for j in range(0, n):
                    correction[j][i] = 0;
            else:
                for j in range(0, n):
                    correction[j][i] = 1/n;
        correction = np.array(correction)
        matrix = matrix + correction
        return matrix
    else:
        return



adjacency_matrix = []
num_nodes = int(input("Enter the number of nodes: "))
```

```
print("Enter the adjacency matrix:")
for _ in range(num_nodes):
    row = list(map(int, input().split()))
    adjacency_matrix.append(row)
adjacency_matrix = np.array(adjacency_matrix)

personalization_vector = list(map(float, input("Enter the
    personalization vector: ").split()))
personalization_vector = np.array(personalization_vector)

adjacency_matrix2=adjacency_matrix.copy()



num_nodes1 = len(adjacency_matrix)
while num_nodes1>0:
    if np.any(np.all(adjacency_matrix == 0, axis=0)):
        adjacency_matrix=check_and_modify_stochastic(
            adjacency_matrix)
    num_nodes1=num_nodes1-1


pagerank = personalized_pagerank(adjacency_matrix,
    personalization_vector)
print("Final probabality of each:-",pagerank)
num_nodes = len(adjacency_matrix)
i=0
sum=0
while i<num_nodes:
    sum=sum+pagerank[i]
    i=i+1

print("Total Probabality:-",sum)


pagerank2=pagerank.copy()




pagerank=np.concatenate((pagerank, np.full(17-num_nodes, -1)))
WebPages = {
    "https://www.wikipedia.org/": pagerank[0],
    "https://www.apple.com" : pagerank[1],
    "https://www.youtube.com": pagerank[2],
    "https://www.facebook.com": pagerank[3],
    "https://www.amazon.com": pagerank[4],
    "https://www.twitter.com" : pagerank[5],
    "https://www.instagram.com" : pagerank[6],
    "https://www.linkedin.com" : pagerank[7],
    "https://www.pinterest.com" : pagerank[8],
    "https://www.snapchat.com" : pagerank[9],
    "https://www.reddit.com" : pagerank[10],
    "https://www.netflix.com" : pagerank[11],
    "https://www.spotify.com" : pagerank[12],
    "https://www.github.com" : pagerank[13],
    "https://www.microsoft.com" : pagerank[14],
    "https://www.google.com": pagerank[15],
    " https://www.cnn.com" : pagerank[16],
}
num_nodes = len(adjacency_matrix)
sorted_webpages = dict(sorted(WebPages.items(), key=lambda x: x[1],
     reverse=True))


print("Webpage order:-")
i=0
for key, value in sorted_webpages.items():
    if i<num_nodes:
        print(i+1,key, value)
    i=i+1



adjacency_matrix_Trans = np.transpose(adjacency_matrix2)
graph = nx.DiGraph(adjacency_matrix_Trans)
labels = {i: f"{i}\n{pagerank[i]}" for i in range(len(pagerank2))}
nx.draw(graph, with_labels=True, labels=labels, node_color='
    lightblue', node_size=500, font_size=12, edge_color='gray',
    arrows=True)
plt.show()
plt.savefig("figure.png")
```

8 PageRank with Personalization

Personalization allows users to receive search results that are tailored to their preferences.

8.1 Personalization in PageRank

To incorporate personalization into PageRank, we introduce a personalized vector V. This vector represents the user's preferences and biases towards specific pages. The personalized vector modifies the calculation of the PageRank scores as follows:

$$PR(p) = [\frac{1-d}{N} + d \sum_{i \in M(p)} \frac{PR(i)}{L(i)}] \cdot V_p$$

where:

- V_p is the value in the personalized vector V for page p.

By multiplying the PageRank score with the corresponding value from the personalized vector, the algorithm incorporates the user's preferences when ranking pages.

8.2 Calculating Personalized PageRank

To calculate the personalized PageRank, we first need to obtain the basic PageRank scores using the original algorithm. Then, we multiply each PageRank score with the corresponding value from the personalized vector.

Let $PR_{\text{basic}}(p)$ be the basic PageRank score of page p and V_p be the value in the personalized vector V for page p. The personalized PageRank score, denoted as $PR_{\text{personalized}}(p)$, can be calculated as:

$$PR_{\text{personalized}}(p) = PR_{\text{basic}}(p) \cdot V_p$$

Then we normalizes the elements of this $\text{PR}_{\text{personalized}}Vector$

9 Further points

9.1 Google's Web Search Algorithm

Google PageRank is a fundamental component of Google's web search algorithm. While the exact details of Google's algorithm are proprietary and constantly evolving, the following information provides an overview of how PageRank is typically used in web search:

9.1.1 Crawling and Indexing:

Google's web crawler (known as Googlebot) systematically visits web pages and follows links to discover new content. During the crawling process, Googlebot collects information about web pages, including their URLs, content, and outgoing links. The collected information is stored in Google's index, which forms the foundation for search results.

9.1.2 PageRank Calculation:

Google calculates the PageRank values for web pages in its index using an iterative algorithm based on the original PageRank formulation, as already described in detail above.

9.1.3 Query Processing:

When a user enters a search query, Google's search engine analyzes the query to understand its intent and identify relevant web pages. The search engine retrieves a subset of web pages from its index that are deemed potentially relevant to the query.

9.1.4 Ranking and Displaying Results:

Google's search engine ranks the retrieved web pages based on a combination of factors, including PageRank, relevance to the query, and other relevance signals. PageRank plays a crucial role in determining the initial ranking order of search results.

Web pages with higher PageRank scores are generally considered more important and are more likely to appear higher in the search results.

9.1.5 Additional Ranking Factors:

Google's search algorithm incorporates various additional factors, such as content relevance, user signals (e.g., click-through rates, dwell time), freshness, and contextual relevance. These factors are used to refine and personalize the search results based on the user's query and search history.

10 Conclusion

The Google PageRank project, developed by Larry Page and Sergey Brin, revolutionized the field of web search and became a fundamental component of Google's search engine. The project aimed to measure the importance and relevance of web pages based on their link structure, effectively creating a ranking system that helped deliver more accurate search results.

The PageRank algorithm was not only significant for its impact on search engine technology but also for its influence on the broader field of network analysis and graph theory. It introduced the concept of using network connectivity to assess the importance of nodes (web pages) within a network (the World Wide Web).

Over the years, PageRank has evolved and been refined with various improvements and extensions. Researchers have explored topic-sensitive PageRank, personalized PageRank, and efficient computation methods to handle the growing scale of the web. These advancements have enhanced the accuracy, personalization, and efficiency of the algorithm.

In conclusion, the Google PageRank project has had a profound impact on the field of web search and network analysis. Its innovative approach to ranking web pages based on their link structure laid the foundation for more sophisticated algorithms and sparked a multitude of research and advancements in the field. PageRank's influence extends beyond search engines, making it a landmark contribution to the study of networks and connectivity.

11 Contributions

- **Divyarajsinh Mahida(2022101085):** Studied and analysed Buildup of google PageRank, its Background Mathematics and described concepts of Linear Algebra involved. Worked on “Developing a formula to rank pages(Simple solution and Alternative)”. Also studied and added state of the art literature to the project.

- **Harsh Gupta (2022101067):** Studied and Analysed the implications of the damping factor and dangling nodes in controlling the flow of importance and ensuring the effectiveness of the PageRank algorithm. Worked on how the damping factor affects PageRank calculations. Designed a working simulation of PageRank. Made the Powerpoint presentation.

- **Samyak Mishra (2022101121) :** Studied and analysed about the connection between the web graph's link structure and the relevance ratings given by PageRank. Worked on the section “The Actual PageRank Algorithm”. Described the iterative Method to calculate PageRank Scores, with its examples and “Dangling Nodes”.

References

[1] Brin, S., Page, L. (1998). The Anatomy of a Large-Scale Hypertextual Web Search Engine. *Computer Networks and ISDN Systems*, 30(1-7), 107-117.

[2] Page, L., Brin, S., Motwani, R., Winograd, T. (1999). The PageRank Citation Ranking: Bringing Order to the Web. *Technical Report, Stanford InfoLab.*

[3] MATHEMATICS BEHIND GOOGLE'S PAGERANK ALGORITHM BY BRIAN MOOR

[4] The PageRank Algorithm By Cesar O. Aguilar

[5] ” The $25,000,000,000* Eigenvector: The Linear Algebra Behind Google Kurt Bryan† and Tanya Leise‡ ”