PES University, Bangalore

(Established under Karnataka Act No. 16 of 2013)

UE21MA141B- LINEAR ALGEBRA AND ITS APPLICATIONS

LAA- PROJECT

Session: Jan-May 2023

Branch :COMPUTER SCIENCE

Semester & Section : 4-B

Sl No.	Name of the Student	SRN	Marks Allotted (Out of 5)
1.	ARAVILLI ATCHUTA RAM	PES2UG21CS086	
2.	ARYAN WADHWA	PES2UG21CS098	
3.	ATHARVA MENKUDLE	PES2UG21CS104	
4.	ANIRUDH BHATTA J.A	PES2UG21CS071	

Name of the Course Instructor :

Signature of the Course Instructor

(with Date) : _____

Title: Eigen Vectors in Google PageRank Algorithm: A Basic Implementation

Abstract

This project focuses on the implementation of eigen vectors in the Google PageRank algorithm, which is widely used to rank web pages based on their importance.

The algorithm is based on the concept that a page's importance is determined by the importance of the pages that link to it.

By representing the web as a graph and using the eigen vector corresponding to the largest eigenvalue of a matrix derived from the graph, the algorithm calculates the relative importance of each page.

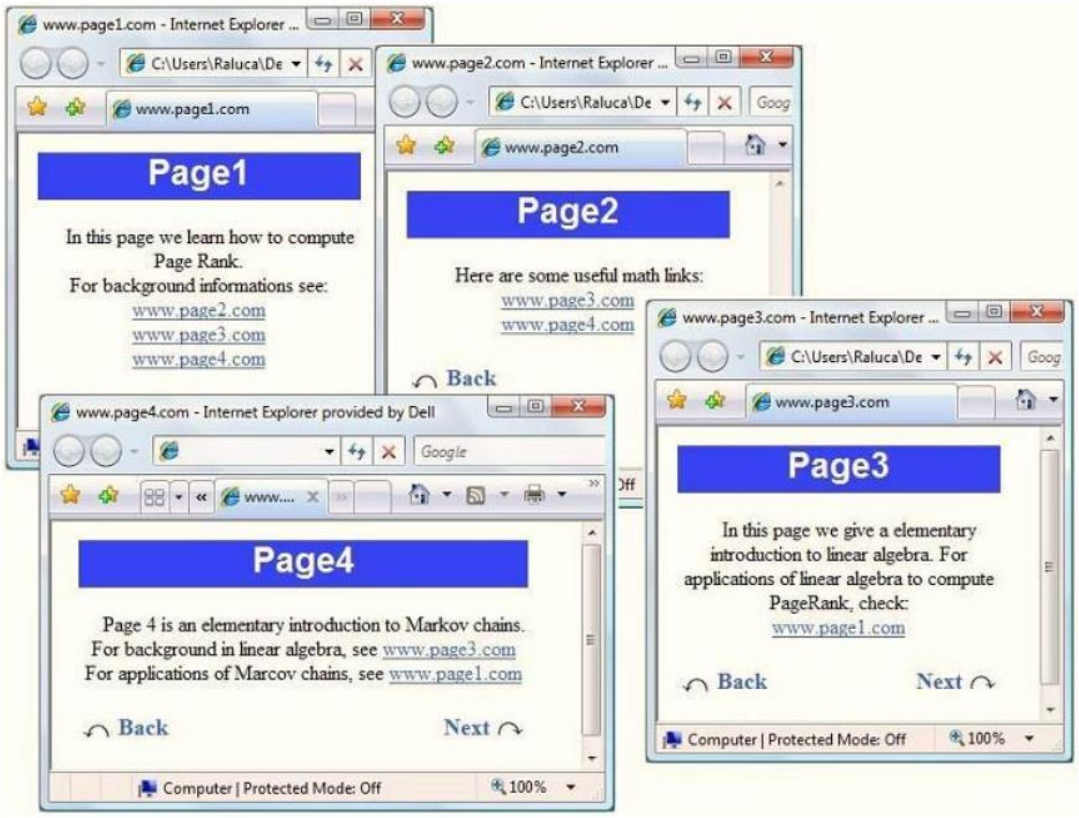
In this project, we provide a detailed explanation of the basic implementation of the PageRank algorithm using eigen vectors.

We also discuss how Google uses the eigenvector to determine the ranking of web pages in its search results.

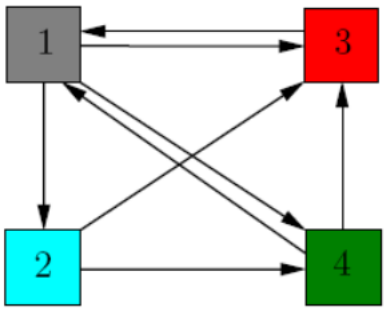
Finally, we evaluate the effectiveness of the algorithm on a sample web graph and demonstrate its ability to provide accurate rankings.

Method

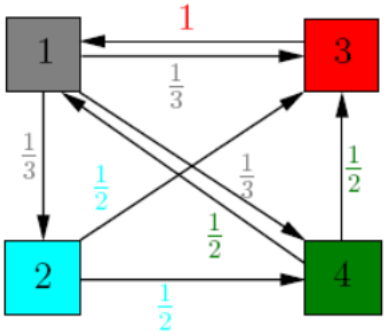
We consider a micro internet with just 4 websites www.page1.com, www.page2.com, www.page3.com, www.page4.com, referencing each other in the manner suggested by the picture



We "translate" the picture into a directed graph with 4 nodes, one for each web site. When web site *i* references *j*, we add a directed edge between node *i* and node *j* in the graph. For the purpose of computing their page rank, we ignore any navigational links such as back, next buttons, as we only care about the connections between different web sites. For instance, Page1 links to all of the other pages, so node 1 in the graph will have outgoing edges to all of the other nodes. Page3 has only one link, to Page 1, therefore node 3 will have one outgoing edge to node 1. After analyzing each web page, we get the following graph:



In our model, each page should transfer evenly its importance to the pages that it links to. Node 1 has 3 outgoing edges, so it will pass on $\frac{1}{3}$ of its importance to each of the other 3 nodes. Node 3 has only one outgoing edge, so it will pass on all of its importance to node 1. In general, if a node has *k* outgoing edges, it will pass on $\frac{1}{k}$ of its importance to each of the nodes that it links to. Let us better visualize the process by assigning weights to each edge.



Let us denote by *A* the transition matrix of the graph, $A = \begin{bmatrix} 0 & 0 & 1 & \frac{1}{2} \\ \frac{1}{3} & 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{3} & \frac{1}{2} & 0 & 0 \end{bmatrix}$.

Finding the pagerank

There are 2 methods to find the pagerank

1. Dynamic system

Dynamical systems point of view:

Suppose that initially the importance is uniformly distributed among the 4 nodes, each getting ¼. Denote by v the initial rank vector, having all entries equal to ¼. Each incoming link increases the importance of a web page, so at step 1, we update the rank of each page by adding to the current value the importance of the incoming links. This is the same as multiplying the matrix A with v . At step 1, the new importance vector is $v_1 = Av$. We can iterate the process, thus at step 2, the updated importance vector is $v_2 = A(Av) = A^2v$. Numeric computations give:

$$v = \begin{pmatrix} 0.25 \\ 0.25 \\ 0.25 \\ 0.25 \end{pmatrix}, \quad Av = \begin{pmatrix} 0.37 \\ 0.08 \\ 0.33 \\ 0.20 \end{pmatrix}, \quad A^2 v = A(Av) = A \begin{pmatrix} 0.37 \\ 0.08 \\ 0.33 \\ 0.20 \end{pmatrix} = \begin{pmatrix} 0.43 \\ 0.12 \\ 0.27 \\ 0.16 \end{pmatrix}$$

$$A^3 v = \begin{pmatrix} 0.35 \\ 0.14 \\ 0.29 \\ 0.20 \end{pmatrix}, \quad A^4 v = \begin{pmatrix} 0.39 \\ 0.11 \\ 0.29 \\ 0.19 \end{pmatrix}, \quad A^5 v = \begin{pmatrix} 0.39 \\ 0.13 \\ 0.28 \\ 0.19 \end{pmatrix}$$

$$A^6 v = \begin{pmatrix} 0.38 \\ 0.13 \\ 0.29 \\ 0.19 \end{pmatrix}, \quad A^7 v = \begin{pmatrix} 0.38 \\ 0.12 \\ 0.29 \\ 0.19 \end{pmatrix}, \quad A^8 v = \begin{pmatrix} 0.38 \\ 0.12 \\ 0.29 \\ 0.19 \end{pmatrix}$$

We notice that the sequences of iterates $v, Av, \ldots, A^k v$ tends to the equilibrium value $v^* = \begin{pmatrix} 0.38 \\ 0.12 \\ 0.29 \\ 0.19 \end{pmatrix}$. We call this the PageRank vector of our web graph.

2. Linear Algebra

Let us denote by x_1, x_2, x_3, and x_4 the importance of the four pages. Analyzing the situation at each node we get the system:

$$\begin{cases} x_1 = 1 \cdot x_3 + \frac{1}{2} \cdot x_4 \\ x_2 = \frac{1}{3} \cdot x_1 \\ x_3 = \frac{1}{3} \cdot x_1 + \frac{1}{2} \cdot x_2 + \frac{1}{2} \cdot x_4 \\ x_4 = \frac{1}{3} \cdot x_1 + \frac{1}{2} \cdot x_2 \end{cases}$$

This is equivalent to asking for the solutions of the equations $A \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$.

Since A is a column stochastic matrix, 1 is an eigen vector automatically.

Hence

The eigenvector $\frac{1}{31} \cdot \begin{bmatrix} 12 \\ 4 \\ 9 \\ 6 \end{bmatrix} \sim \begin{bmatrix} 0.38 \\ 0.12 \\ 0.29 \\ 0.19 \end{bmatrix}$ is our PageRank vector.

Code:

HTML Parser

```
from bs4 import BeautifulSoup
from numpy import linalg as LA

import numpy as np
 ## Give a list of all HTML files , list of strings
def trans_mat(l):
    s1=set(l)
    mat_a=[]
    for i in l:
        HTMLFile = open(i, "r")
        index = HTMLFile.read()

        soup = BeautifulSoup(index,'lxml')
```

```
        d={}
        s=set()
        for a in soup.find_all('a', href=True):
            s.add(a['href'])
        ##print("Found the URL:", a['href'])
            if(a['href'] not in d):
                d[a['href']]=0
            d[a['href']]+=1
        # print(s)
        # print(s1)
        s3=(s1.difference(s))
        for l in s3:
            d[l]=0
        n=sorted(d)
        ans=[]
        for index in n:
            ans.append(d[index])

        #ans=[round(i/sum(ans),2) for i in ans]
        ans=[i/sum(ans) for i in ans]
        mat_a.append(ans)
        #print(mat_a)

    n=np.array(mat_a)
    k=np.transpose(n)
    ##print(k)
    b=k.tolist()
    return b




l=["page1.html","page2.html","page3.html","page4.html"]

p=trans_mat(l)
print(p)
```

Dynamical Systems Point of View

```
##Page Rank vector from dynamical systems point of view

#Example
## A=[[0,0,1,1/2],[1/3,0,0,0],[1/3,1/2,0,1/2],[1/3,1/2,0,0]]
import numpy as np
def pr(a,n):
    v=[1/n for i in range(0,n)]
```

```
    prod=np.dot(a,v)
    print("A.V is =\n")
    print(prod)
    for i in range(1,10):
        prod=np.dot(a,prod) ##
        print("A^{}.V is=".format(i))
        print(prod)
A=[[0,0,1,1/2],[1/3,0,0,0],[1/3,1/2,0,1/2],[1/3,1/2,0,0]]
pr(A,4)
```

Output:

```
PS C:\Users\atchu\Documents\PES_EC\LA_project> python pr_d.py
A.V is =

[0.375      0.08333333 0.33333333 0.20833333]
A^1.V is=
[0.4375     0.125      0.27083333 0.16666667]
A^2.V is=
[0.35416667 0.14583333 0.29166667 0.20833333]
A^3.V is=
[0.39583333 0.11805556 0.29513889 0.19097222]
A^4.V is=
[0.390625   0.13194444 0.28645833 0.19097222]
A^5.V is=
[0.38194444 0.13020833 0.29166667 0.19618056]
A^6.V is=
[0.38975694 0.12731481 0.29050926 0.19241898]
A^7.V is=
[0.38671875 0.12991898 0.28978588 0.19357639]
A^8.V is=
[0.38657407 0.12890625 0.29065394 0.19386574]
A^9.V is=
[0.38758681 0.12885802 0.29024402 0.19331115]
```

Linear Algebra Point of View

```
##Page Rank vector from dynamical systems point of view

#Example
## A=[[0,0,1,1/2],[1/3,0,0,0],[1/3,1/2,0,1/2],[1/3,1/2,0,0]]
import numpy as np
def pr(a,n):
    v=[1/n for i in range(0,n)]
    prod=np.dot(a,v)
    print("A.V is =\n")
    print(prod)
    for i in range(1,10):
```

```
        prod=np.dot(a,prod) ##
        print("A^{}.V is=".format(i))
        print(prod)
A=[[0,0,1,1/2],[1/3,0,0,0],[1/3,1/2,0,1/2],[1/3,1/2,0,0]]
pr(A,4)
```

Output:

```
PS C:\Users\atchu\Documents\PES_EC\LA_project> python eig.py
Eigenvectors corresponding to the eigenvalue 1:
[0.38709677+0.j 0.12903226+0.j 0.29032258+0.j 0.19354839+0.j]
```

References:

http://pi.math.cornell.edu/~mec/Winter2009/RalucaRemus/Lecture3/lecture3.html

PageRank - Wikipedia