

## Experiment 8 - Implementation of Page Rank algorithm

**Name:** Shaikh Shadab

**Rollno :** 17DCO74

**Class :** TE.CO

**Batch :** B3

### #Theory

#### ➤ Page Rank (PR)

PageRank is a link analysis algorithm and it assigns a numerical weighting to each element of a hyperlinked set of documents, such as the World Wide Web, with the purpose of "measuring" its relative importance within the set. The algorithm may be applied to any collection of entities with reciprocal quotations and references. The numerical weight that it assigns to any given element  $E$  is referred to as the PageRank of  $E$  and denoted by  $PR(E)$ . Other factors like Author Rank can contribute to the importance of an entity.

#### ➤ A PageRank results from a mathematical algorithm based on the webgraph, created by all World Wide Web pages as nodes and hyperlinks as edges, taking into consideration authority hubs such as cnn.com or usa.gov. The rank value indicates an importance of a particular page. A hyperlink to a page counts as a vote of support. The PageRank of a page is defined recursively and depends on the number and PageRank metric of all pages that link to it ("incoming links"). A page that is linked to by many pages with high PageRank receives a high rank itself.

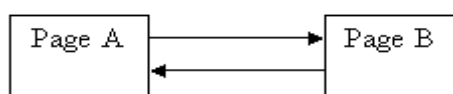
#### ➤ In short PageRank is a "vote", by all the other pages on the Web, about how important a page is. A link to a page counts as a vote of support. If there's no link there's no support (but it's an abstention from voting rather than a vote against the page).

#### ➤ Working of PR

PR assumes page  $A$  has pages  $T_1...T_n$  which point to it (i.e., are citations). The parameter  $d$  is a damping factor which can be set between 0 and 1. PR usually set  $d$  to 0.85. Also  $C(A)$  is defined as the number of links going out of page  $A$ . The PageRank of a page  $A$  is given as follows:

$$PR(A) = (1-d) + d (PR(T_1)/C(T_1) + \dots + PR(T_n)/C(T_n))$$

PageRanks form a probability distribution over web pages, so the sum of all web pages' PageRanks will be one. PageRank or  $PR(A)$  can be calculated using a simple iterative algorithm, and corresponds to the principal eigenvector of the normalized link matrix of the web. What that means to us is that we can just go ahead and calculate a page's PR without knowing the final value of the PR of the other pages. That seems strange but, basically, each time we run the calculation we're getting a closer estimate of the final value. So all we need to do is remember the each value we calculate and repeat the calculations lots of times until the numbers stop changing much. Lets take the simplest example network: two pages, each pointing to the other:



Each page has one outgoing link (the outgoing count is 1, i.e.  $C(A) = 1$  and  $C(B) = 1$ ).

➤ Key terminologies

- i.  $PR(T_n)$  - Each page has a notion of its own self-importance. That's " $PR(T_1)$ " for the first page in the web all the way up to " $PR(T_n)$ " for the last page
- ii.  $C(T_n)$  - Each page spreads its vote out evenly amongst all of its outgoing links. The count, or number, of outgoing links for page 1 is " $C(T_1)$ ", " $C(T_n)$ " for page n, and so on for all pages.
- iii.  $PR(T_n)/C(T_n)$  - so if our page (page A) has a backlink from page "n" the share of the vote page A will get is " $PR(T_n)/C(T_n)$ "
- iv.  $d(\dots$  - All these fractions of votes are added together but, to stop the other pages having too much influence, this total vote is "damped down" by multiplying it by 0.85 (the factor "d")
- v.  $(1 - d)$  - The  $(1 - d)$  bit at the beginning is a bit of probability math magic so the "sum of all web pages' PageRanks will be one": it adds in the bit lost by the  $d(\dots$ . It also means that if a page has no links to it (no backlinks) even then it will still get a small PR of 0.15 (i.e.  $1 - 0.85$ ). (Aside: the Google paper says "the sum of all pages" but they mean the "the normalised sum" – otherwise known as "the average" to you and me.
- vi. Damping factor - The PageRank theory holds that an imaginary surfer who is randomly clicking on links will eventually stop clicking. The probability, at any step, that the person will continue is a damping factor d. Various studies have tested different damping factors, but it is generally assumed that the damping factor will be set around 0.85.[5] In applications of PageRank to biological data, a Bayesian analysis finds the optimal value of d to be 0.31. The damping factor is subtracted from 1 (and in some variations of the algorithm, the result is divided by the number of documents (N) in the collection) and this term is then added to the product of the damping factor and the sum of the incoming PageRank scores. That is,

$$PR(A) = \frac{1 - d}{N} + d \left( \frac{PR(B)}{L(B)} + \frac{PR(C)}{L(C)} + \frac{PR(D)}{L(D)} + \dots \right).$$

When calculating PageRank, pages with no outbound links are assumed to link out to all other pages in the collection. Their PageRank scores are therefore divided evenly among all other pages. In other words, to be fair with pages that are not sinks, these random transitions are added to all nodes in the Web. This residual probability, d, is usually set to 0.85, estimated from the frequency that an average surfer uses his or her browser's bookmark feature. So, the equation is as follows:

$$PR(p_i) = \frac{1 - d}{N} + d \sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(p_j)}$$

Where  $p_1, p_2, \dots, p_n$  are the pages under consideration,  $M(p_i)$  is the set of pages that link to  $p_i$ ,  $L(p_j)$  is the number of outbound links on page  $p_j$ , and  $N$  is the total number of pages.

➤ Eigen Matrix R

The PageRank values are the entries of the dominant right eigenvector of the modified adjacency matrix rescaled so that each column adds up to one. This makes PageRank a particularly elegant metric: the eigenvector is

$$\mathbf{R} = \begin{bmatrix} PR(p_1) \\ PR(p_2) \\ \vdots \\ PR(p_N) \end{bmatrix}$$

where  $\mathbf{R}$  is the solution of the equation

$$\mathbf{R} = \begin{bmatrix} (1-d)/N \\ (1-d)/N \\ \vdots \\ (1-d)/N \end{bmatrix} + d \begin{bmatrix} \ell(p_1, p_1) & \ell(p_1, p_2) & \cdots & \ell(p_1, p_N) \\ \ell(p_2, p_1) & \ddots & & \vdots \\ \vdots & & \ell(p_i, p_j) & \\ \ell(p_N, p_1) & \cdots & & \ell(p_N, p_N) \end{bmatrix} \mathbf{R}$$

where the adjacency function  $\ell(p_i, p_j)$  is the ratio between number of links outbound from page  $j$  to page  $i$  to the total number of outbound links of page  $j$ . And link is 0 if page  $p_j$  does not link to  $p_i$ , and normalized such that, for each  $j$

$$\sum_{i=1}^N \ell(p_i, p_j) = 1,$$

➤ Computing iteration

At  $t = 0$ , an initial probability distribution is assumed, usually

$$PR(p_i; 0) = \frac{1}{N}.$$

where  $N$  is the total number of pages, and  $p_i; 0$  is page  $i$  at time 0.

At each time step, the computation, as detailed above, yields

$$PR(p_i; t+1) = \frac{1-d}{N} + d \sum_{p_j \in M(p_i)} \frac{PR(p_j; t)}{L(p_j)}$$

where  $d$  is the damping factor,

or in matrix notation

$$\mathbf{R}(t+1) = d\mathcal{M}\mathbf{R}(t) + \frac{1-d}{N}\mathbf{1}, (*)$$

where  $\mathbf{R}_i(t) = PR(p_i; t)$  and  $\mathbf{1}$  is the column vector of length  $N$  containing only ones.

The matrix  $\mathcal{M}$  is defined as

$$\mathcal{M}_{ij} = \begin{cases} 1/L(p_j), & \text{if } j \text{ links to } i \\ 0, & \text{otherwise} \end{cases}$$

i.e.,

$$\mathcal{M} := (K^{-1}A)^T,$$

where  $A$  denotes the adjacency matrix of the graph and  $K$  is the diagonal matrix with the outdegrees in the diagonal.

The probability calculation is made for each page at a time point, then repeated for the next time point. The computation ends when for some small  $\epsilon$

$$|\mathbf{R}(t+1) - \mathbf{R}(t)| < \epsilon,$$

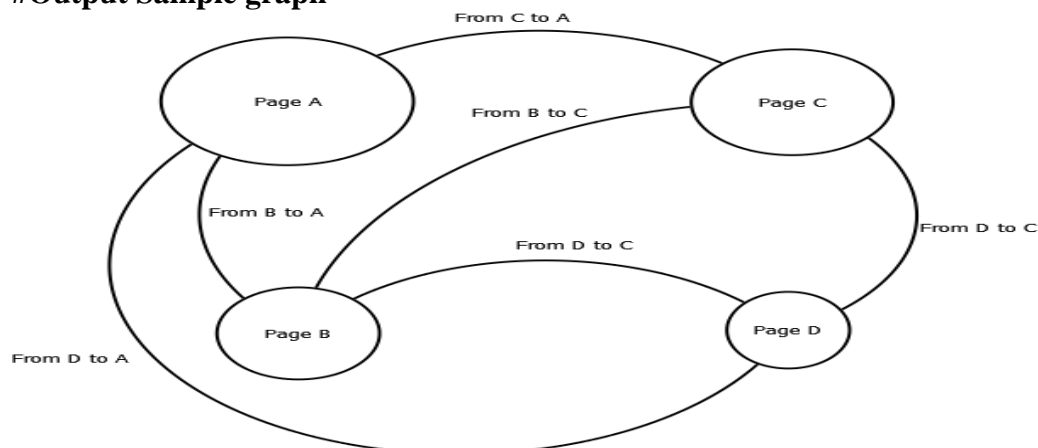
i.e., when convergence is assumed.

Assume,  $\epsilon = 1.08e-8$

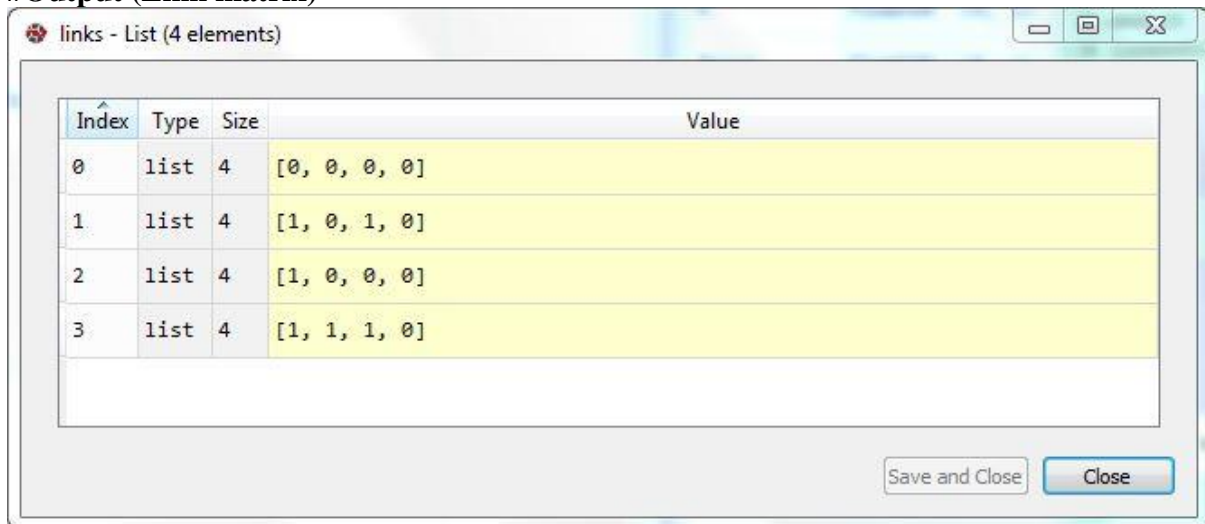
## #Source Code

```
import numpy as np
pageno=int(input("Enter total number of pages"))
d=0.85 #damping factor
eps=1.08e-8 #use for convergence
links=[]
for i in range(0,pageno):
    L=[]
    for j in range(0,pageno):
        L.append(int(input("Type 1 if there is a link from page '+str(i+1)+' To page '+str(j+1)")))
    links.append(L)
outBound=[]
count=0
for i in range(len(links)):
    for j in range(len(links[i])):
        if(links[i][j]==1):
            count+=1
        if(j==len(links[i])-1):
            outBound.append(count)
    count=0
M=np.zeros((pageno,pageno))
for i in range(0,pageno):
    for j in range(0,pageno):
        if(links[j][i]==1):
            M[i][j]=1/outBound[j]
M=np.matrix(M)
onceColMat=np.matrix(np.ones((pageno,1),dtype=int))
R=np.matrix(np.full((pageno,1),1/pageno))
while True:
    Rnext = d*np.dot(M,R)+((1-d)/pageno)*onceColMat
    diff = np.subtract(Rnext,R)
    if np.linalg.norm(diff)<eps:
        break
    else:
        R=Rnext
```

## #Output Sample graph



### #Output (Link matrix)

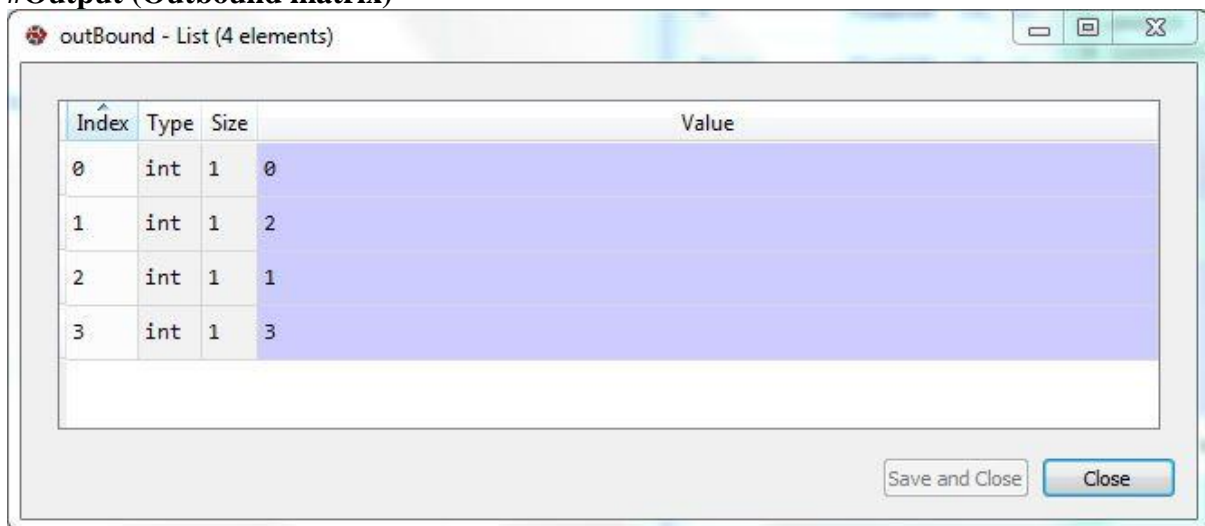


links - List (4 elements)

Index	Type	Size	Value
0	list	4	[0, 0, 0, 0]
1	list	4	[1, 0, 1, 0]
2	list	4	[1, 0, 0, 0]
3	list	4	[1, 1, 1, 0]

Save and Close Close

### #Output (Outbound matrix)



outBound - List (4 elements)

Index	Type	Size	Value
0	int	1	0
1	int	1	2
2	int	1	1
3	int	1	3

Save and Close Close

### #Output (M Matrix)



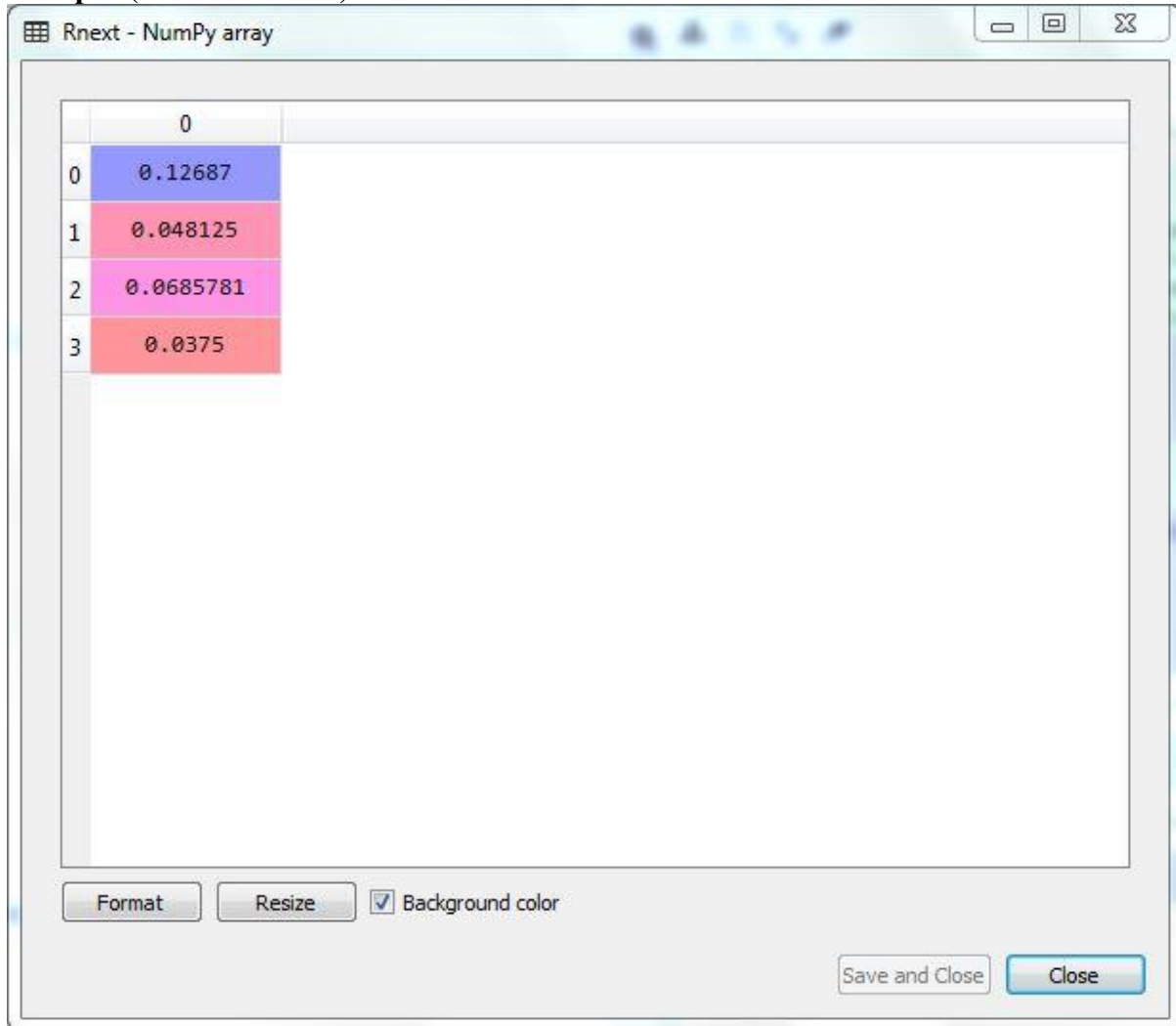
M - NumPy array

	0	1	2	3
0	0	0.5	1	0.333333
1	0	0	0	0.333333
2	0	0.5	0	0.333333
3	0	0	0	0

Format Resize ☒ Background color

Save and Close Close

### #Output (Final R Matrix)



	0
0	0.12687
1	0.048125
2	0.0685781
3	0.0375

### #Conclusion

In this experiment we have seen the initial Page Rank algorithm which was used by google for assigns a numerical weighting to each element of a hyperlinked set of documents, such as the World Wide Web, with the purpose of "measuring" its relative importance within the set. We have also seen various computation strategies and key terminologies involved in this algorithm