Implementation of Network Ranking using PageRank and Random Walk in Python

Ahmad Farhan AlShammari
Department of Computer and Information Systems
College of Business Studies, PAAET
Kuwait

ABSTRACT
The goal of this research is to implement network ranking using PageRank and random walk in Python. Network ranking is used to calculate the importance of nodes in the network. It helps to order the nodes according to their ranking values. Network ranking is performed using PageRank and random walk. The final results are compared to make sure that the two methods are matching.

The basic steps of network ranking using PageRank and random walk are explained: defining network (nodes, adjacency matrix, and rank vector), computing outgoing nodes, computing transition matrix, performing PageRank, performing random walk, comparing results, and plotting charts.

The developed program was tested on an experimental data. The program has successfully performed the basic steps of network ranking using PageRank and random walk and provided the required results.

Keywords
Computer Science, Artificial Intelligence, Machine Learning, Network Ranking, PageRank, Random Walk, Python, Programming.

1. INTRODUCTION
In the recent years, machine learning has played a major role in the development of computer systems. Machine learning (ML) is a branch of Artificial Intelligence (AI) which is focused on the study of algorithms and methods to improve the performance and efficiency of computer programs [1-10].

Network ranking is an important area in machine learning. It is sharing knowledge with many other fields like: programming, data science, mathematics, statistics, and numerical methods [11-14, 15-19].

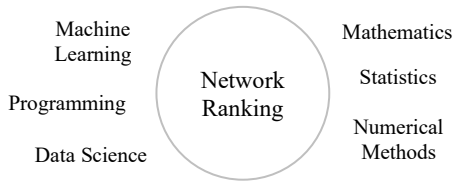


Fig 1: Area of Network Ranking

Network ranking is used to measure the importance of nodes in the network. It helps to order the nodes according to their ranking values. It is performed using PageRank and random walk. Network ranking is widely used in many applications especially for searching and filtering.

2. LITERATURE REVIEW
The literature was reviewed to explore the fundamental concepts, methods, and applications of network ranking using PageRank and random walk [20-25, 26-31, 32-34].

Actually, networks are very essential in life. They exist everywhere connecting many things like: computers, mobiles, students, friends, articles, posts, tweets, words, etc.

Network ranking is an important area in machine learning. It helps to determine the importance of nodes in the network. It has a wide range of applications in different fields: technology, business, education, sociology, environment, sports, etc.

In this research, network ranking is performed using two methods: PageRank and random walk.

PageRank was developed by Larry Page and Sergey Brin in 1998. It was used to order the web pages in the World-Wide Web. Later, Page and Brin founded the internet giant, Google [35]. Now, PageRank is widely used in different applications with several variations.

Random walk is used to simulate the behavior of the network. It assumes a "random surfer" that moves between nodes based on their probabilities. After a long run, the network will reach equilibrium and will converge to the stationary form. Random walk was first applied by Andrey Markov to prove his theory on Markov chains to understand random processes for large numbers [36].

The fundamental concepts of network ranking using PageRank and random walk are explained in the following section.

Network Ranking:
Network ranking is the process of calculating the importance of nodes in the network. The nodes are ordered according to their ranking values. It is basically used in searching and filtering.

Network ranking is performed using two methods: PageRank and random walk.

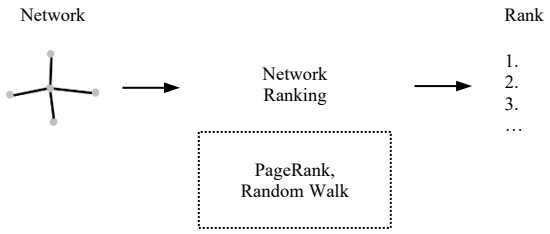


Fig 2: Concept of Network Ranking

Network:

The network is a set of nodes connected by edges. Networks exist everywhere in life. For example, the traffic system is a network where cities are the nodes and routes are the edges. In general, networks can be physical, logical, social, biological, etc. They may include cities, people, computers, web pages, articles, elements, processes, etc.

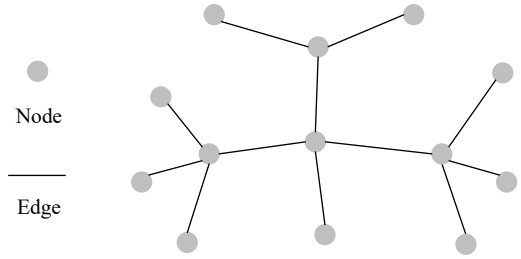


Fig 3: Concept of Network

Types of Networks:

The networks are divided into two types: directed and undirected.

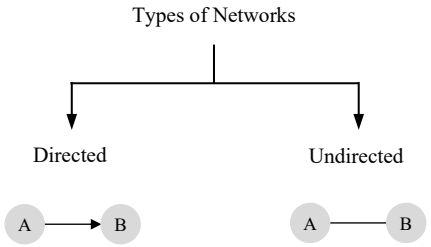


Fig 4: Types of Networks

In directed network, the edge is drawn as an arrow, where the flow goes in one direction (from *A* to *B*). While, in undirected network, the edge is drawn as a line, where the flow goes in both directions (from *A* to *B* and from *B* to *A*).

Transition Diagram:

The transition diagram is a graph that represents the network. The nodes are drawn as circles and the edges are drawn as arrows (directed) or lines (undirected). The following transition diagram represents a complete directed network:

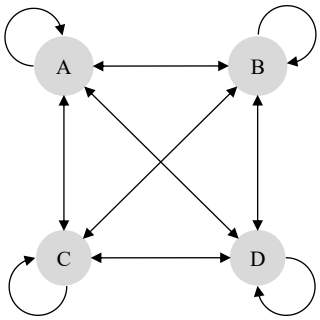


Fig 5: Representation of Transition Diagram

Adjacency Matrix:

The adjacency matrix is a matrix of size ($n \times n$), where (n) is the number of nodes. It can be represented as shown in the following form:

$$A = \begin{array}{c} \\ 0 \\ 1 \\ \vdots \\ i \\ \vdots \\ n\text{-}1 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & \dots & j & \dots & n\text{-}1 \end{array} \\ \begin{bmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,j} & \cdots & a_{0,n-1} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,j} & \cdots & a_{1,n-1} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{i,0} & a_{i,1} & \cdots & a_{i,j} & \cdots & a_{i,n-1} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n-1,0} & a_{n-1,1} & \cdots & a_{n-1,j} & \cdots & a_{n-1,n-1} \end{bmatrix} \end{array}$$

Where:

$$a_{i,j} = \begin{cases} 1, & \text{If there is edge} \\ 0, & \text{otherwise} \end{cases}$$

Transition Matrix:

The transition matrix is a matrix of size ($n \times n$) that shows the probabilities of transition between nodes. It can be represented as shown in the following form:

$$P = \begin{array}{c} \\ 0 \\ 1 \\ \vdots \\ i \\ \vdots \\ n\text{-}1 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & \dots & j & \dots & n\text{-}1 \end{array} \\ \begin{bmatrix} p_{0,0} & p_{0,1} & \cdots & p_{0,j} & \cdots & p_{0,n-1} \\ p_{1,0} & p_{1,1} & \cdots & p_{1,j} & \cdots & p_{1,n-1} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ p_{i,0} & p_{i,1} & \cdots & p_{i,j} & \cdots & p_{i,n-1} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ p_{n-1,0} & p_{n-1,1} & \cdots & p_{n-1,j} & \cdots & p_{n-1,n-1} \end{bmatrix} \end{array}$$

Where ($p_{i,j}$) is the probability of transition from node (i) to node (j). The transition matrix should satisfy the following conditions:

$$1 \geq p_{i,j} \geq 0$$

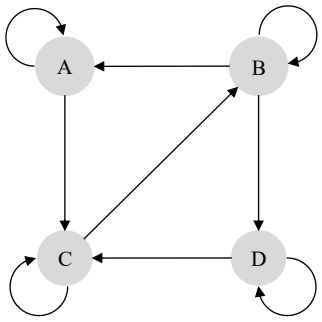
and

$$\sum_{j} p_{i,j} = 1$$

Where, the probability is always in the range [0, 1], and the sum of probabilities in each row is (1).

Example:

Assume a network with the following transition diagram:



Then, the adjacency matrix is shown in the following form:

$$A = \begin{array}{c} \\ A \\ B \\ C \\ D \end{array} \begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \end{array}$$

And the transition matrix is shown in the following form:

$$P = \begin{array}{c} \\ A \\ B \\ C \\ D \end{array} \begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{bmatrix} 1/2 & 0 & 1/2 & 0 \\ 1/3 & 1/3 & 0 & 1/3 \\ 0 & 1/2 & 1/2 & 0 \\ 0 & 0 & 1/2 & 1/2 \end{bmatrix} \end{array}$$

PageRank:

PageRank is a famous algorithm used to order nodes according to their ranking values. The rank of a node is affected by the rank of the linking nodes and the number of outgoing edges.

The basic formula of PageRank algorithm is shown here:

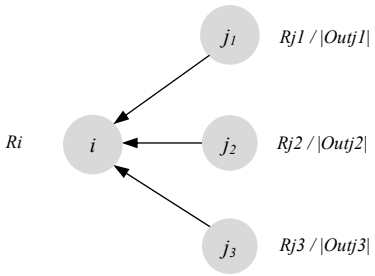
$$R_i = \alpha . \sum_{j \to i} \frac{R_j}{|Out_j|} + \frac{(1 - \alpha)}{N} \qquad (1)$$

Where:

(R_j) is the rank of the linking node (j),
($|Out_j|$) is the number of outgoing edges in node (j),
(α) is the damping factor, and
(N) is the number of nodes.

Explanation:

The rank of a given node (i) is computed by dividing the rank of the linking nodes (j_1, j_2, j_3) by the number of outgoing edges. Then, the calculated sum ($R_{j1} / |Out_{j1}| + R_{j2} / |Out_{j2}| + R_{j3} / |Out_{j3}|$) is multiplied by the damping factor (α) and added to the value ($1 - \alpha$) /N.



$$Rj = \alpha . (Rj1 / |Outj1| + Rj2 / |Outj2| + Rj3 / |Outj3|) + (1 - \alpha) / N$$
$$\therefore R = \alpha . (R . P) + (1 - \alpha) / N$$

Fig 6: Explanation of PageRank

Using Markov chains, the rank vector (R) is computed for each iteration as shown in the following steps:

$$R_1 = \alpha . (R_0 . P) + \frac{(1 - \alpha)}{N}$$

$$R_2 = \alpha . (R_1 . P) + \frac{(1 - \alpha)}{N}$$

$$R_3 = \alpha . (R_2 . P) + \frac{(1 - \alpha)}{N}$$

$$\dots$$

$$R_{n+1} = \alpha . (R_n . P) + \frac{(1 - \alpha)}{N} \qquad (2)$$

After a long run, the rank vector (R) will be stable and will not change. This indicates that the rank vector is converging to the stationary form.

Here, the implementation of PageRank to compute the stationary rank vector (R) is explained step by step in the following algorithm:

Algorithm 1: PageRank

```
# define nodes
nodes = […]
# define adjacency matrix
A = [[…],
      … ,
     […]]
# define rank vector
R = […]
# compute outgoing nodes
Out = compute_Out(A)
# compute transition matrix
T = compute_T(A, Out)
# start with R
R_old = R
# initialize distance
D = []
# number of iterations
N = 10^6
for t = 1 to N do
    # compute R_new
    R_new = alpha*multiply(R_old, T) + (1 – alpha)/n
    print(t, R_new)
    # compute distance
    d = distance(R_new, R_old)
    # add distance
    D.append(d)
    # check if equal
    if (R_new = R_old) then
        break
    # make R_new as R_old
    R_old = R_new
end for
R = R_new
```

Random Walk:

The random walk is a simulation method used to compute the stationary rank vector. It assumes a random surfer that keeps moving from one node to another.

After a long run, the rank vector will be stable and will not change. This indicates that the rank vector is converging to the stationary form.

Here, the implementation of random walk to compute the stationary rank vector (R) is explained step by step in the following algorithm:

Algorithm 2: Random Walk

```
# define nodes
nodes = […]
# define adjacency matrix
A = [[…],
      … ,
     […]]
# define rank vector
R = […]
# compute outgoing nodes
Out = compute_Out(A)
# start with node
current_node = …
# initialize walk_path
walk_path = [current_node]
# number of iterations
N = 10^6
for t = 1 to N do
```

```
        # select random number
        r = random.random()
        # alpha:
        if (r < alpha) then
            # no outgoing nodes
            if (|Out[current_node]| = 0) then
                # select node from nodes
                next_node = random.choice(nodes)
            # outgoing nodes
            else
                # select node from Out nodes
                next_node = random.choice(Out[current_node])
        # 1-alpha:
        else
            # select node from nodes
            next_node = random.choice(nodes)
        # update node count
        R[next_node] += 1
        # add node to walk path
        walk_path.append(next_node)
        # make next_node as current_node
        current_node = next_node
end for
# normalize R
R = normalize(R)
print(R)
```

Network Ranking System:

The network ranking system is briefly described in the following overview:

Input: Network.
Output: Rank.
Processing: First, the network is defined (nodes, adjacency matrix, and rank vector). Then, the outgoing nodes and the transition matrix are computed. Next, the PageRank is performed to compute the stationary rank vector. Also, the random walk is performed to compute the stationary rank vector. After that, the results are compared and the charts are plotted.

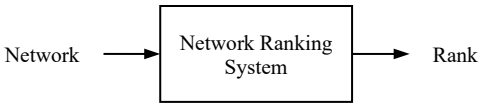


Fig 7: Network Ranking System

Python:

Python [37] is an open source, object-oriented, and general-purpose programming language. It is simple to code, easy to learn, and powerful. It is the most popular programming language, especially in the field of machine learning.

Python provides many additional libraries for different purposes. For example: Numpy [38], Pandas [39], Matplotlib [40], Seaborn [41], SciPy [42], NLTK [43], and SK Learn [44].

3. RESEARCH METHODOLOGY

The basic steps of network ranking using PageRank and random walk are: (1) defining network (nodes, adjacency matrix, and rank vector), (2) computing outgoing nodes, (3) computing transition matrix, (4) performing PageRank (5) performing random walk, (6) comparing results, and (7) plotting charts.

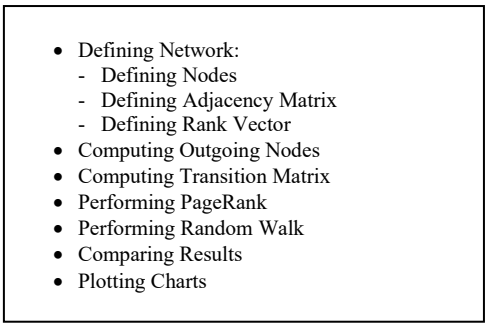


Fig 8: Basic Steps of Network Ranking

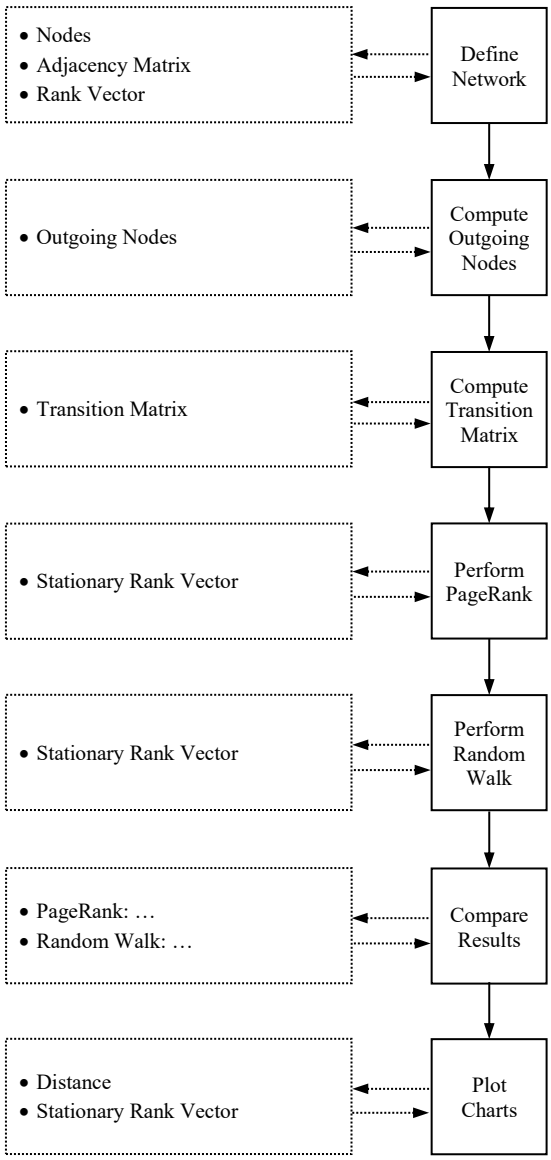


Fig 9: Flowchart of Network Ranking

The basic steps of network ranking using PageRank and random walk are explained in details in the following section.

Note: The program is developed using only the standard functions of Python without any additional library.

1. Defining Network:

The network is defined by the following steps:

1.1. Defining Nodes:

The nodes are defined by the following code:

```
nodes = [0, 1, ..., n-1]
```

1.2. Defining Adjacency Matrix:

The adjacency matrix (A) is defined by the following code:

```
A = [[a0,0, a0,1, ..., a0,n-1],
     [a1,0, a1,1, ..., a1,n-1],
      ...
     [an-1,0, an-1,1, ..., an-1,n-1]]
```

1.3. Defining Rank Vector:

The initial rank vector (R) is defined by the following code:

```
R = [...]
```

2. Computing Outgoing Nodes:

The outgoing nodes (Out) is computed by the following code:

```
def compute_Out(a):
    n = len(a)
    t = []
    for i in range(n):
        row = []
        for j in range(n):
            if (a[i][j] != 0):
                row.append(j)
        t.append(row)
    return t
```

3. Computing Transition Matrix:

The transition matrix (T) is computed by the following code:

```
def compute_T(a, out):
    n = len(a)
    t = []
    for i in range(n):
        nout = len(out[i])
        if (nout == 0):
            value = n
        else:
            value = nout
        row = []
        for j in range(n):
            row.append(a[i][j]/value)
        t.append(row)
    return t
```

4. Performing PageRank:

The PageRank is performed to compute the stationary rank vector (R) by the following code:

```
# start with R
R_old = R
# distance
D = []
# number of iterations
N = 10**6
for t in range(1, N):
    # compute R_new
    R_new = add(mul(alpha, multiply(R_old, T),
                (1-alpha)/n)
    print(t, ":", R_new)
    # compute distance
    d = distance(R_new, R_old)
    # add distance
    D.append(d)
    # check if equal
    if equal(R_new, R_old):
        break
    # make R_new as R_old
    R_old = R_new
R = R_new
```

The distance between the new and old rank vectors is computed by the following code:

```
def distance(v1, v2):
    sum = 0
    for i in range(len(v1)):
        sum += (v1[i] - v2[i])**2
    return sum**0.5
```

5. Performing Random Walk:

The random walk is performed to compute the stationary rank vector (R) by the following code:

```
import numpy as np

# start with node
current_node = ...
# initialize walk path
walk_path = [current_node]
# number of iterations
N = 10**6
for t in range(1, N):
    # select random number
    r = np.random.random()
    # alpha:
    if (r < alpha):
        # no outgoing nodes
        if (len(Out[current_node]) == 0):
            # select node from nodes
            next_node = np.random.choice(nodes)
        # outgoing nodes
        else:
            # select node from Out nodes
            next_node = np.random.choice(
                        Out[current_node]))
    # 1-alpha:
    else:
        # select node from nodes
        next_node = np.random.choice(nodes)
    # update node count
    R[next_node] += 1
    # add node to walk_path
    walk_path.append(next_node)
    # make next_node as current_node
    current_node = next_node
# normalize R
R = normalize(R)
print("R =", R)
print("Walk Path:")
print(walk_path)
```

To normalize the rank vector (R), the count values are divided by the sum of counts. It is done by the following code:

```
def normalize(v):
    total = sum(v)
    for i in range(len(v)):
        v[i] /= total
    return v
```

6. Comparing Results:

The results of performing PageRank and random walk are printed by the following code:

```
print("(1) PageRank:")
print("R =", R)

print("(2) Random Walk:")
print("R =", R)
```

Then, the results are compared to check if they are matching or not.

7. Plotting Charts:

The distance (*D*) is plotted by the following code:

```
import matplotlib.pyplot as plt

plt.plot(range(len(D)), D, color='red')
plt.title("Distance Plot")
plt.xlabel("Iterations")
plt.ylabel("Value")
plt.show()
```

Also, the rank vector (*R*) is plotted by the following code:

```
plt.bar(nodes, R)
plt.title("Rank Plot")
plt.xlabel("Nodes")
plt.ylabel("Value")
plt.show()
```

4. RESULTS AND DISCUSSION

The developed program was tested on an experimental data. The program has successfully performed the basic steps of network ranking using PageRank and random walk and provided the required results. The program output is explained in details in the following section.

Defining Network:

The network is defined by the following steps:

1. Defining Nodes:

The nodes are defined and printed as shown in the following view:

```
Nodes = [0, 1, 2, 3]
```

2. Defining Adjacency Matrix:

The adjacency matrix (*A*) is defined and printed as shown in the following view:

```
Adjacency Matrix (A):
[0, 1, 1, 0]
[0, 0, 1, 0]
[1, 0, 0, 0]
[0, 0, 1, 0]
```

3. Defining Rank Vector:

The initial rank vector (*R*) is defined and printed as shown in the following view:

```
Rank Vector (R):
[1, 0, 0, 0]
```

Computing Outgoing Nodes:

The outgoing nodes (*Out*) are computed and printed as shown in the following view:

```
Outgoing Nodes (Out):
0: [1, 2]
1: [2]
2: [0]
3: [2]
```

Computing Transition Matrix:

The transition matrix (*T*) is computed and printed as shown in the following view:

```
Transition Matrix (T):
[0.0, 0.5, 0.5, 0.0]
[0.0, 0.0, 1.0, 0.0]
[1.0, 0.0, 0.0, 0.0]
[0.0, 0.0, 1.0, 0.0]
```

Performing PageRank:
Computing Stationary Rank Vector:

The PageRank is performed to compute the rank vector (*R*) for each iteration. It is printed as shown in the following view:

```
Performing PageRank ...
Rank Vector (R):
 1: [0.0375, 0.4625, 0.4625, 0.0375]
 2: [0.430625, 0.053438, 0.478438, 0.037500]
 3: [0.444172, 0.220516, 0.297813, 0.037500]
 4: [0.290641, 0.226273, 0.445586, 0.037500]
 5: [0.416248, 0.161022, 0.385229, 0.037500]
 6: [0.364945, 0.214406, 0.383149, 0.037500]
 7: [0.363177, 0.192602, 0.406721, 0.037500]
 8: [0.383213, 0.191850, 0.387437, 0.037500]
 9: [0.366821, 0.200366, 0.395313, 0.037500]
10: [0.373516, 0.193399, 0.395585, 0.037500]
...
```

The stationary rank vector (*R*) is computed and printed as shown in the following view:

```
R = [0.372527, 0.195824, 0.394149, 0.037500]
```

Performing Random Walk:
Computing Stationary Rank Vector:

The random walk is performed to compute the rank vector (*R*) for each iteration. It is printed as shown in the following view:

```
Performing Random Walk ...
Rank Vector (R):
 1: [1, 0, 1, 0]
 2: [2, 0, 1, 0]
 3: [2, 0, 1, 1]
 4: [2, 0, 2, 1]
 5: [3, 0, 2, 1]
 6: [3, 1, 2, 1]
 7: [3, 1, 3, 1]
 8: [4, 1, 3, 1]
 9: [4, 1, 4, 1]
10: [5, 1, 4, 1]
...
```

The stationary rank vector (*R*) is computed and printed as shown in the following view:

```
R = [0.372226, 0.195795, 0.394429, 0.037550]
```

The walk path is computed and printed as shown in the following view:

```
Walk Path:
[0, 2, 0, 3, 2, 0, 1, 2, 0, 2, 0, 2, 0, 1, ...
```

Comparing Results:

The results of performing PageRank and random walk are printed as shown in the following view:

```
(1) PageRank:
R = [0.372527, 0.195824, 0.394149, 0.037500]

(2) Random Walk:
R = [0.372226, 0.195795, 0.394429, 0.037550]
```

The comparison shows that the results of performing PageRank and random walk are matching.

Plotting Charts:

The distance (D) is computed and plotted as shown in the following chart:

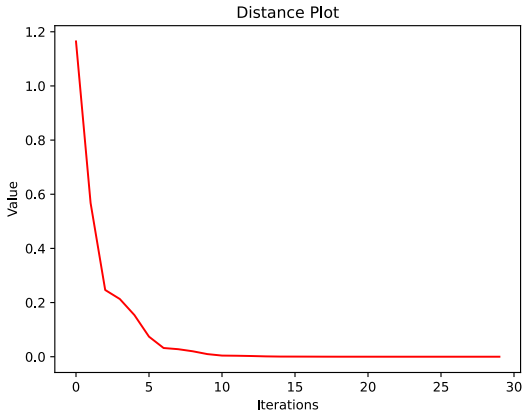


Fig 10: Distance Plot

The plot shows that the distance is decreasing with iterations. This indicates that the rank vector (R) is converging to the stationary form.

Now, the stationary rank vector (R) using PageRank is plotted as shown in the following chart:

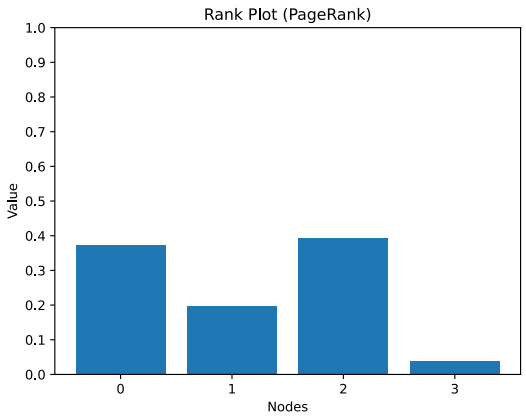


Fig 11: Rank Plot using PageRank

The plot shows the importance of nodes. They are sorted by their ranking values in the following order (2, 0, 1, 3).

The stationary rank vector (R) using random walk is plotted as shown in the following chart:

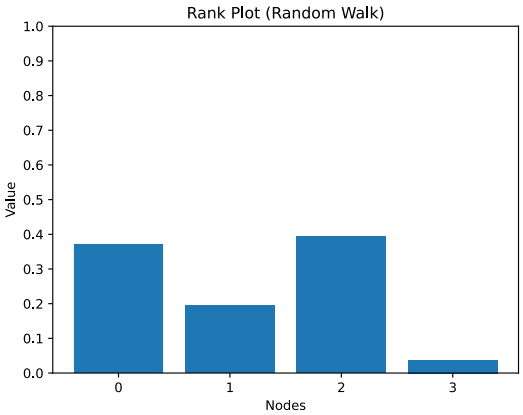


Fig 12: Rank Plot using Random Walk

The plot shows that random walk has provided the same result as in PageRank.

The nodes are listed in descending order according to their ranking values as shown in the following table:

Table 1: Listing Nodes by Rank in Descending Order

Order	Node	Rank (%)
1	2	39.42
2	0	37.25
3	1	19.58
4	3	3.75
		100 %

In summary, it is clear that the program has successfully performed the basic steps of network ranking using PageRank and random walk and provided the required results.

5. CONCLUSION

In this research, the goal was to implement network ranking using PageRank and random walk in Python. The literature was reviewed to explore the fundamental concepts of network ranking using PageRank and random walk: network ranking, network, types of networks, transition diagram, adjacency matrix, transition matrix, PageRank, and random walk.

The author developed a program in Python to perform the basic steps of network ranking using PageRank and random walk: defining network (nodes, adjacency matrix, and rank vector), computing outgoing nodes, computing transition matrix, performing PageRank, performing random walk, comparing results, and plotting charts.

The developed program was tested on an experimental data. The program has successfully performed the basic steps of network ranking using PageRank and random walk and provided the required results.

In the future, more work is needed to improve the current methods of network ranking using PageRank and random walk. In addition, they should be more investigated on different fields and domains.

6. REFERENCES

[1] Sammut, C., & Webb, G. I. (2011). "Encyclopedia of Machine Learning". Springer.

[2] Jung, A. (2022). "Machine Learning: The Basics". Springer.

[3] Kubat, M. (2021). "An Introduction to Machine Learning". Springer.

[4] Li, H. (2023). "Machine Learning Methods". Springer.

[5] Zollanvari, A. (2023). " Machine Learning with Python". Springer.

[6] Chopra, D., & Khurana, R. (2023). "Introduction to Machine Learning with Python". Bentham Science Publishers.

[7] Müller, A. C., & Guido, S. (2016). "Introduction to Machine Learning with Python: A Guide for Data Scientists". O'Reilly Media.

[8] Raschka, S. (2015). "Python Machine Learning". Packt Publishing.

[9] Forsyth, D. (2019). "Applied Machine Learning". Springer.

[10] Sarkar, D., Bali, R., & Sharma, T. (2018). "Practical Machine Learning with Python". Apress.

[11] Igual, L., & Seguí, S. (2017). "Introduction to Data Science: A Python Approach to Concepts, Techniques and Applications". Springer.

[12] VanderPlas, J. (2017). "Python Data Science Handbook: Essential Tools for Working with Data". O'Reilly Media.

[13] Muddana, A., & Vinayakam, S. (2024). "Python for Data Science". Springer.

[14] Unpingco, J. (2021). "Python Programming for Data Analysis". Springer.

[15] Zelle, J. (2017). "Python Programming: An Introduction to Computer Science". Franklin, Beedle & Associates.

[16] Xanthidis, D., Manolas, C., Xanthidou, O. K., & Wang, H. I. (2022). "Handbook of Computer Programming with Python". CRC Press.

[17] Chun, W. (2001). "Core Python Programming". Prentice Hall Professional.

[18] Padmanabhan, T. (2016). "Programming with Python". Springer.

[19] Beazley, D., & Jones, B. K. (2013). "Python Cookbook: Recipes for Mastering Python 3". O'Reilly Media.

[20] Newman, M. (2018). "Networks: An Introduction". Oxford University Press.

[21] Estrada, E. & Knight, P. (2015). "A First Course in Network Theory". Oxford University Press.

[22] Menczer, F., Fortunato, S., & Davis, C. A. (2020). "A First Course in Network Science". Cambridge University Press.

[23] Barabasi, A. (2016). "Network Science". Cambridge University Press

[24] Lewis, T. (2009). "Network Science: Theory and Applications". John Wiley & Sons.

[25] Knickerbocker, D. (2023). "Network Science with Python". Packt Publishing.

[26] Norris, J. (2009). "Markov Chains". Cambridge University Press.

[27] Tolver, A. (2016). "An Introduction to Markov Chains". Department of Mathematical Sciences, University of Copenhagen.

[28] Weber, R. (2011). "Markov Chains". Department of Pure Mathematics and Mathematical Statistics. University of Cambridge.

[29] Ching, W., Huang, S., Ng, M., & Siu, T. (2013). "Markov Chains: Models, Algorithms, and Applications". Springer.

[30] Privault, N. (2018). "Understanding Markov Chains: Examples and Applications". Springer.

[31] Ankan, A. & Panda, A. (2018). "Hands-On Markov Models with Python". Packt Publishing.

[32] 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.

[33] Page, L., Brin, S., Motwani, R., & Winograd, T. (1999). "The PageRank Citation Ranking: Bringing Order to The Web". Stanford University.

[34] Langville, A., & Meyer, C. (2011). "Google's PageRank and Beyond: The Science of Search Engine Rankings". Princeton University Press.

[35] Google: http://www.google.com

[36] Grinstead, C., & Snell, J. (1997). "Introduction to Probability". American Mathematical Society.

[37] Python: http://www.python.org

[38] Numpy: http://www.numpy.org

[39] Pandas: http://pandas.pydata.org

[40] Matplotlib: http://www. matplotlib.org

[41] Seaborn: http://seaborn.pydata.org

[42] SciPy: http://scipy.org

[43] NLTK: http://www.nltk.org

[44] SK Learn: http://scikit-learn.org