LAB Manual

PART A

(PART A: TO BE REFFERED BY STUDENTS)

Experiment No.10

A.1 Aim:

Implementation of Page rank/HITS algorithm

A.2 Prerequisite:

Familiarity programming languages.

A.3 Outcome:

After successful completion of this experiment students will be able to

Explore and Implement web mining

A.4 Theory:

THEORY:

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). Quoting from the original Google paper, PageRank is defined like this:

We assume page A has pages T1...Tn which point to it (i.e., are citations). The parameter d is a damping factor which can be set between 0 and 1. We usually set d to 0.85. There are more details about d in the next section. 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(T1)/C(T1) + ... + PR(Tn)/C(Tn))

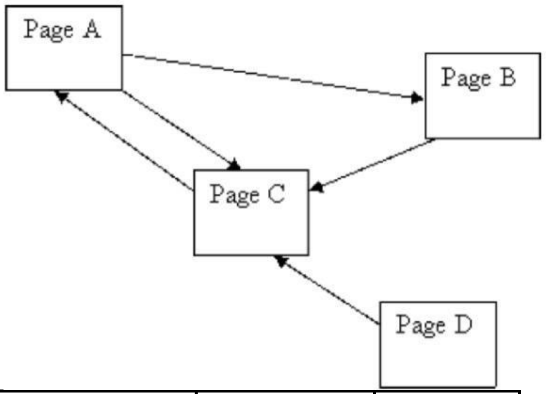
Note that the PageRanks form a probability distribution over web pages, so the sum of all web pages' PageRanks will be one. PageRank or PR(A) can be calculated using a simple iterative algorithm, and corresponds to the principal eigenvector of the normalized link matrix of the web. but that’s not too helpful so let’s break it down into sections.

1. **PR(Tn) -** Each page has a notion of its own self-importance. That’s “PR(T1)” for the first page in the web all the way up to “PR(Tn)” for the last page
2. **C(Tn) -** Each page spreads its vote out evenly amongst all of it’s outgoing links. The count, or number, of outgoing links for page 1 is “C(T1)”, “C(Tn)” for page n, and so on for all pages.
3. **PR(Tn)/C(Tn) -** so if our page (page A) has a backlink from page “n” the share of the vote page A will get is “PR(Tn)/C(Tn)”
4. **d(... -** 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”)

5. **(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(** 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.

How is PageRank Calculated?

The PR of each page depends on the PR of the pages pointing to it. But we won't know what PR those pages have until the pages pointing to **them** have their PR calculated and so on… And when you consider that page links can form circles it seems impossible to do this calculation! But actually it's not that bad. Remember this bit of the Google paper:
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.



Node/page	Out link	Incoming link
A	2	1
B	1	1
C	1	3
D	1	0

PR(A) = 1-0.85 + 0.85(1/1) = 0.15+0.85 =1
PR(B) = 1-085 + 0.85(1/2) = 0.575
PR(C) = 1-0.85 + 0.85(1/2+0.575/1+1/1) = 0.15 + 0.85*2.5 = 1.91375
PR(D) = 0.15 + 0.85*0 = 0.15
PR = C,A,B,D
K=2
PR(A) = 0.15 + 0.85 (1.91375/1) = 1.7766875
PR(B) = 0.15 + 0.85(1.7766875/2) = 0.9050921875
PR(C) = 0.15 + 0.85(1.7766875/2+0.9050921875/1+0.15/1) = 1.8019
PR(D) = 0.15
PR = C,A,B,D

PART B

(PART B: TO BE COMPLETED BY STUDENTS)

(Students must submit the soft copy as per following segments within two hours of the practical. The soft copy must be uploaded on the Blackboard or emailed to the concerned lab in charge faculties at the end of the practical in case the there is no Black board access available)

Roll. No. A12	Name: SUFIYAN KHAN
Class: TE AI& DS	Batch: A1
Date of Experiment:	Date of Submission:
Grade:	

B.1 Software Code written by student:

HTM algorithm:

```
class HTM:
    def __init__(self, input_size, output_size):
        self.weights = np.random.rand(output_size, input_size)

    def forward(self, input_data):
        return np.dot(self.weights, input_data)

# Example usage:
import numpy as np

input_data = np.array([0.2, 0.4, 0.1, 0.8, 0.5])
input_size = len(input_data)
output_size = 3

htm = HTM(input_size, output_size)
output = htm.forward(input_data)
print("HTM output:", output)
```

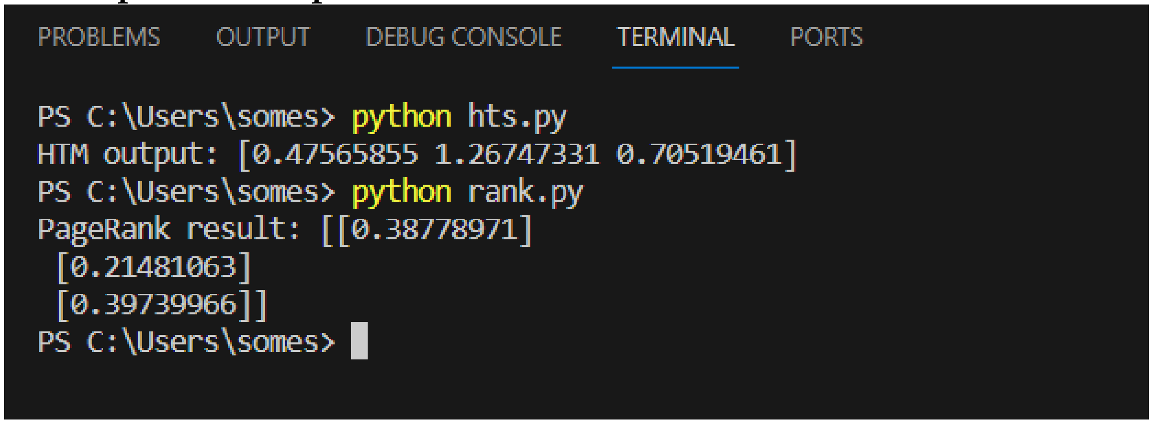
Page Rank Algorithm:

```
import numpy as np
def pagerank(M, num_iterations=100, d=0.85):
  N = M.shape[1]
  v = np.random.rand(N, 1)
  v = v / np.linalg.norm(v, 1)
  M_hat = (d * M) + (((1 - d) / N) * np.ones((N, N)))

  for i in range(num_iterations):
    v = M_hat @ v
  return v
# Example usage:
M = np.array([[0, 0, 1],
       [0.5, 0, 0],
       [0.5, 1, 0]])

result = pagerank(M, num_iterations=100, d=0.85)
print("PageRank result:", result)
```

B.2 Input and Output:



```
PS C:\Users\somes> python hts.py
HTM output: [0.47565855 1.26747331 0.70519461]
PS C:\Users\somes> python rank.py
PageRank result: [[0.38778971]
 [0.21481063]
 [0.39739966]]
PS C:\Users\somes>
```

B.3 Observations and learning:

These implementations serve as introductory examples of these algorithms, showcasing their fundamental principles and providing a starting point for further exploration and understanding.

B.4 Conclusion:

We have successfully Implemented both the algorithms And learned all the basic techniques

B.5 Question of Curiosity

Q1: Advantages and Disadvantages of Agglomeration and Hierarchical Clustering:

Agglomeration Clustering:

Advantages:
1. Simplicity: Agglomeration clustering is relatively simple to implement and understand, making it a popular choice for many applications.
2. Scalability: It can be applied to large datasets with minimal computational requirements, making it scalable for big data applications.
3. Works with various distance measures: Agglomeration clustering can work with different distance measures, allowing for flexibility in analyzing different types of data.
4. Can handle non-globular shapes: It can identify clusters with non-globular shapes, making it useful for complex data patterns.

Disadvantages:
1. High computational cost for large datasets: Although it is generally scalable, the computational cost can still become significant for very large datasets.
2. Sensitivity to noise and outliers: Agglomeration clustering can be sensitive to noise and outliers, leading to suboptimal clustering results.
3. Difficulty in determining the number of clusters: Determining the optimal number of clusters can be challenging in agglomeration clustering.

Hierarchical Clustering:

Advantages:
1. Visualization of results: Hierarchical clustering provides a dendrogram that allows for the visualization of the clustering process, making it easier to interpret the relationships between clusters.
2. No need to specify the number of clusters: Hierarchical clustering does not require the number of clusters to be specified beforehand, which can be advantageous in exploratory data analysis.
3. Flexibility in linkage methods: It can accommodate different linkage methods, such as single linkage, complete linkage, and average linkage, enabling the analysis of various types of data.

Disadvantages:
1. High computational complexity: Hierarchical clustering can be computationally intensive, especially for large datasets, which can limit its scalability.
2. Lack of ability to undo previous steps: Once a decision about the merging of clusters is made, it cannot be undone, which can potentially lead to suboptimal results if the wrong decisions are made early in the process.
3. Difficulty in dealing with noise and outliers: Like agglomeration clustering, hierarchical clustering can also be sensitive to noise and outliers, which can affect the clustering results.

Q2: Relationship Between Top-Down, Bottom-Up, and Division/Agglomeration:

Top-Down (Divisive) and Bottom-Up (Agglomerative) are two different approaches to hierarchical clustering.

Top-Down (Divisive) Approach: This method starts with all the data points as one big cluster and then recursively divides the cluster into smaller subclusters until individual data points are reached.

Bottom-Up (Agglomerative) Approach: This method starts with each data point as a single cluster and then merges the closest clusters step by step until all the data points are merged into one big cluster.

The Division/Agglomeration process is essentially the reverse of each other. In the division process, the clusters are split until each data point is in its own cluster, while in the agglomeration process, individual data points are merged into larger clusters.

Both top-down and bottom-up approaches can be seen as complementary. Top-down helps in understanding the structure of the data by dividing it into smaller subsets, while bottom-up helps in revealing similarities between data points, allowing for the formation of clusters. They represent different strategies for analyzing and understanding the inherent structures within the data.