EX7 Implementation of Link Analysis using HITS Algorithm

DATE: 18:04:24

AIM: To implement Link Analysis using HITS Algorithm in Python.

Description:

The HITS (Hyperlink-Induced Topic Search) algorithm is a link analysis algorithm used to rank web pages. It identifies authority and hub pages in a network of web pages based on the structure of the links between them.

Procedure:

1. ***Initialization:***

   a) Start with an initial set of authority and hub scores for each page.

   b) Typically, initial scores are set to 1 or some random values.

2. ***Construction of the Adjacency Matrix:***

   a) The web graph is represented as an adjacency matrix where each row and column correspond to a web page, and the matrix elements denote the presence or absence of links between pages.

   b) If page A has a link to page B, the corresponding element in the adjacency matrix is set to 1; otherwise, it's set to 0.

3. ***Iterative Updates:***

   a) Update the authority scores based on the hub scores of pages pointing to them and update the hub scores based on the authority scores of pages they point to.

   b) Calculate authority scores as the sum of hub scores of pages pointing to the given page.

   c) Calculate hub scores as the sum of authority scores of pages that the given page points to.

4. ***Normalization:***

   a) Normalize authority and hub scores to prevent them from becoming too large or small.

   b) Normalize by dividing by their Euclidean norms (L2-norm).

5. ***Convergence Check:***

   a) Check for convergence by measuring the change in authority and hub scores between iterations.

b) If the change falls below a predefined threshold or the maximum number of iterations is reached, the algorithm stops.

6. ***Visualization:***

   Visualize using bar chart to represent authority and hub scores.

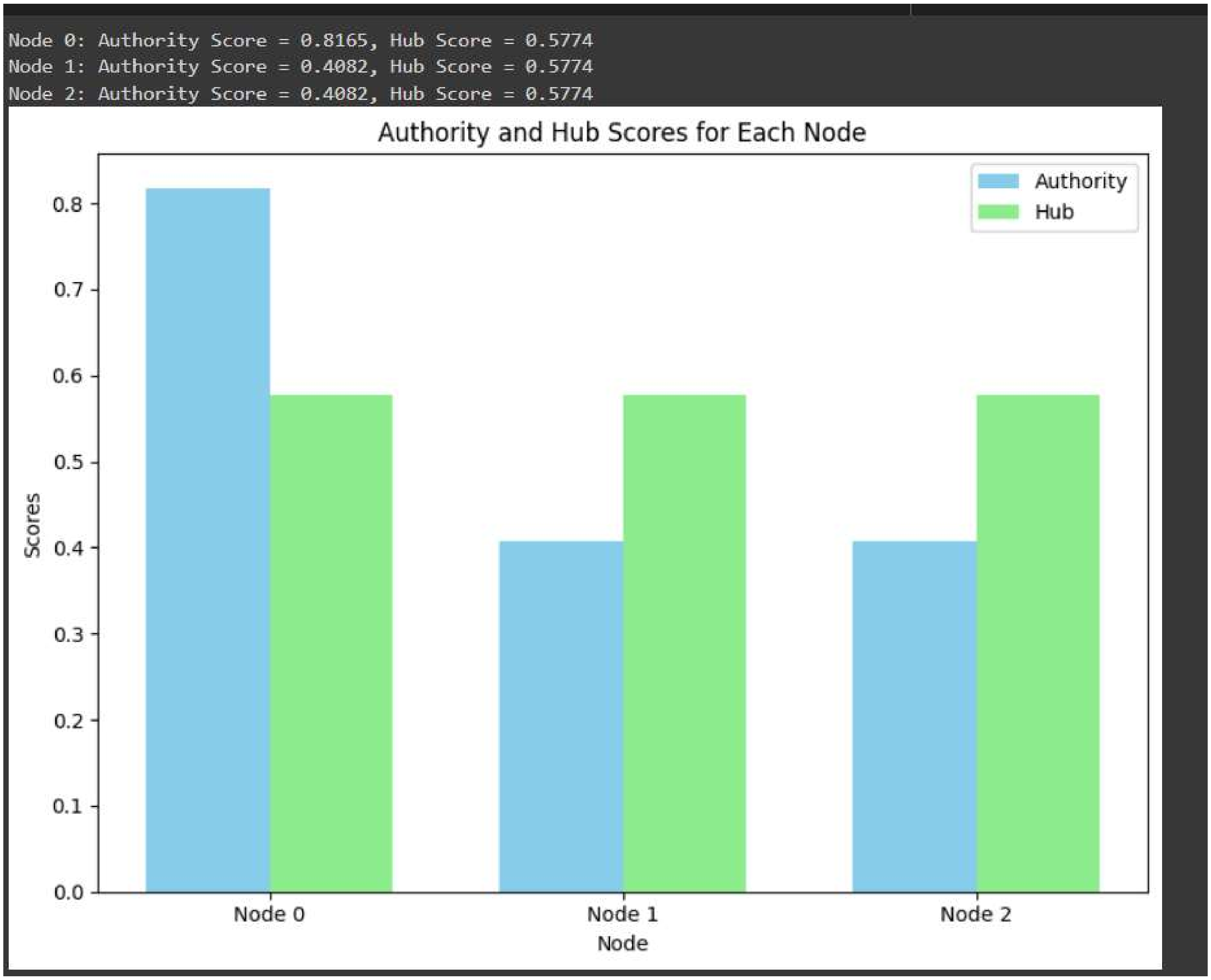
Program:

DEVELOPED BY :S.Mohanraj

REG NO :212221230065

```
import numpy as np
import matplotlib.pyplot as plt

def hits_algorithm(adjacency_matrix, max_iterations=100, tol=1.0e-6):
    num_nodes = len(adjacency_matrix)
    authority_scores = np.ones(num_nodes)
    hub_scores = np.ones(num_nodes)

    for i in range(max_iterations):
        # Authority update
        new_authority_scores = np.dot(adjacency_matrix.T, hub_scores)
        new_authority_scores /= np.linalg.norm(new_authority_scores, ord=2)  # Normalizin

        # Hub update
        new_hub_scores = np.dot(adjacency_matrix, new_authority_scores)
        new_hub_scores /= np.linalg.norm(new_hub_scores, ord=2)  # Normalizing

        # Check convergence
        authority_diff = np.linalg.norm(new_authority_scores - authority_scores, ord=2)
        hub_diff = np.linalg.norm(new_hub_scores - hub_scores, ord=2)

        if authority_diff < tol and hub_diff < tol:
            break

        authority_scores = new_authority_scores
        hub_scores = new_hub_scores

    return authority_scores, hub_scores

# Example adjacency matrix (replace this with your own data)
# For simplicity, using a random adjacency matrix
adj_matrix = np.array([
    [0, 1, 1],
    [1, 0, 0],
    [1, 0, 0]
])

# Run HITS algorithm
authority, hub = hits_algorithm(adj_matrix)
for i in range(len(authority)):
    print(f"Node {i}: Authority Score = {authority[i]:.4f}, Hub Score = {hub[i]:.4f}")

# Bar chart of authority vs hub scores

nodes = np.arange(len(authority))
bar_width = 0.35
plt.figure(figsize=(8, 6))
plt.bar(nodes - bar_width/2, authority, bar_width, label='Authority', color='skyblue')  #
plt.bar(nodes + bar_width/2, hub, bar_width, label='Hub', color='lightgreen')  # Change c
plt.xlabel('Node')
plt.ylabel('Scores')
plt.title('Authority and Hub Scores for Each Node')
plt.xticks(nodes, [f'Node {i}' for i in nodes])
plt.legend()
plt.tight_layout()
plt.show()
```

Output:



```
Node 0: Authority Score = 0.8165, Hub Score = 0.5774
Node 1: Authority Score = 0.4082, Hub Score = 0.5774
Node 2: Authority Score = 0.4082, Hub Score = 0.5774
```

Result:

Thus Link analysis using HITS algorithm in python is successfully implemented