Experiment No 10

Aim: Implementation of Page rank/HITS algorithm

Theory:

PageRank and HITS (Hyperlink-Induced Topic Search) are two important algorithms used in web search and information retrieval to rank web pages. They both aim to determine the importance or relevance of web pages based on their connections and links to other pages. Here's an overview of both algorithms:

1. PageRank Algorithm:

PageRank is an algorithm developed by Larry Page and Sergey Brin, the co-founders of Google. It is used to rank web pages based on the importance of their incoming links. The basic idea behind PageRank is that important pages are more likely to be linked to by other pages.

- PageRank treats the web as a graph, where web pages are nodes, and hyperlinks between them are edges.
- Each page is assigned an initial PageRank value. Typically, all pages are given equal initial PageRank values.
- PageRank is calculated iteratively. At each iteration, each page distributes a fraction of its PageRank to the pages it links to.
- The PageRank of a page is determined by the PageRank of the pages that link to it. A link from a page with high PageRank is more valuable.
- Pages that have more incoming links and are linked from important pages have higher PageRank values.
- The PageRank of a page converges to a stable value after many iterations, and this value represents the page's importance.

Mathematically, the PageRank of a page 'P' is calculated as follows:

$$PR(P) = (1 - d) + d * (PR(Q1)/L(Q1) + PR(Q2)/L(Q2) + ... + PR(Qn)/L(Qn))$$

Where:
- PR(P) is the PageRank of page P.
- d is a damping factor (usually set to 0.85) to account for the probability that a user might randomly jump to any page.
- PR(Q1), PR(Q2), ..., PR(Qn) are the PageRank values of pages that link to page P.
- L(Q1), L(Q2), ..., L(Qn) are the number of outbound links on pages Q1, Q2, ..., Qn.

2. HITS (Hyperlink-Induced Topic Search) Algorithm:

HITS, also known as "Hubs and Authorities," is an algorithm that assesses web page authority and hub status. It was developed by Jon Kleinberg.

- In the HITS algorithm, each web page is assigned two scores: a hub score and an authority score.
- Hub pages are those that link to many authority pages, while authority pages are those linked to by many hub pages.
- The HITS algorithm iteratively updates the hub and authority scores until they converge to stable values.
- The final hub and authority scores provide a way to rank pages based on their role as hubs and authorities in the web graph.

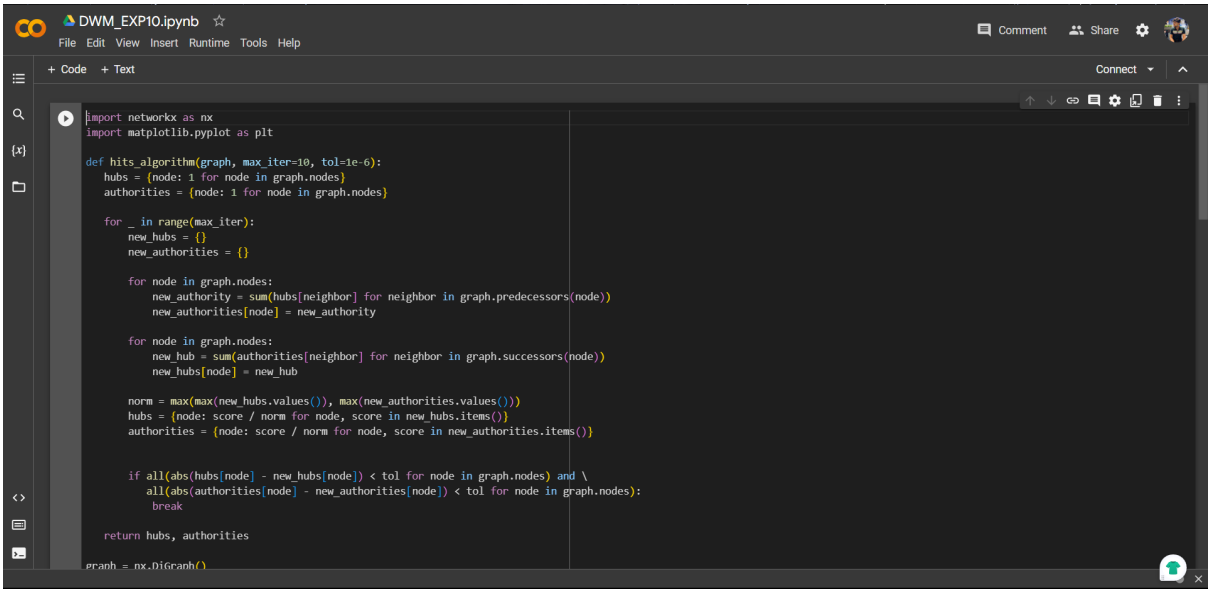
Mathematically, the HITS algorithm works as follows:

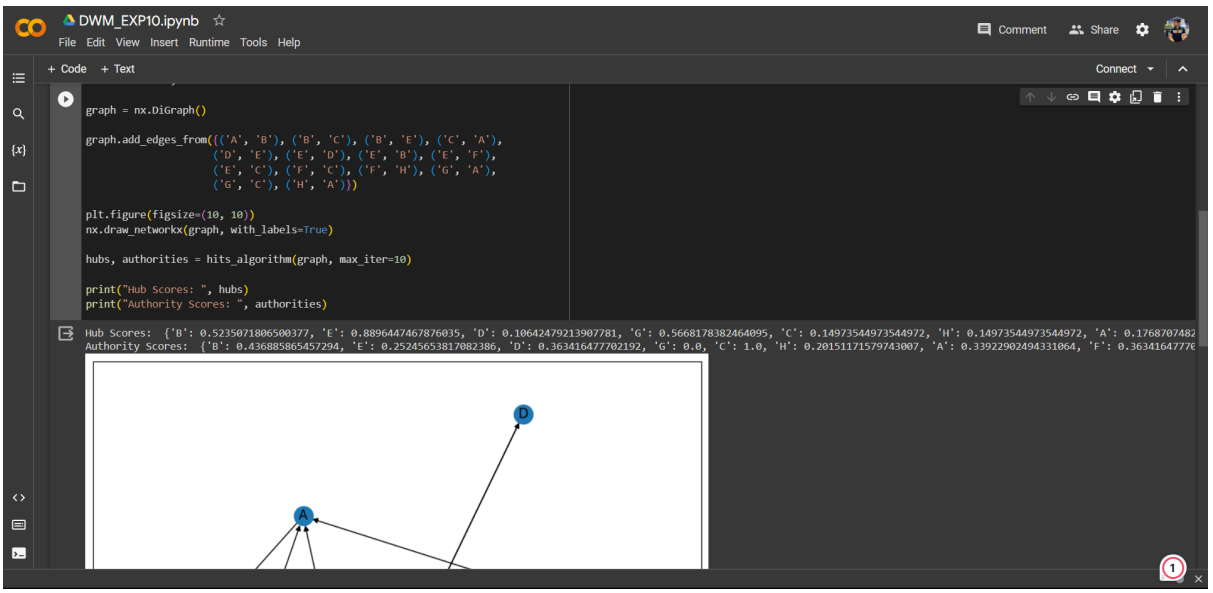
1. Initialize all hub and authority scores to 1.
2. Iteratively update the hub and authority scores based on the links between pages.
3. Normalize the hub and authority scores to ensure they sum to 1.

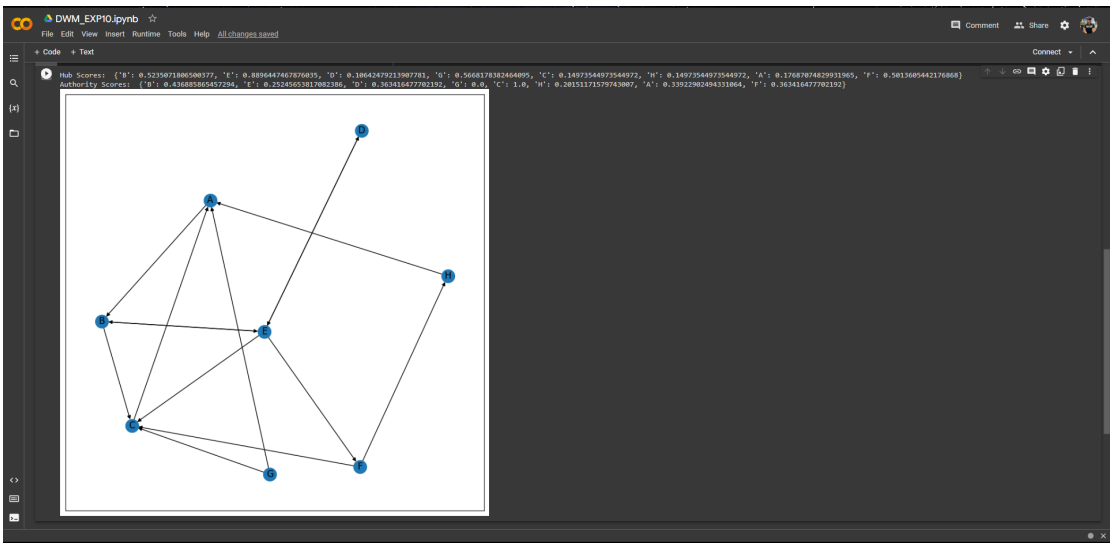
HITS algorithm can be used to find not only the most authoritative pages but also to find good hubs that link to these authoritative pages.

Both PageRank and HITS are foundational algorithms in web search and information retrieval. PageRank is used by Google in its search engine, and HITS provides valuable insights into the link structure of the web. These algorithms have evolved and are part of a broader set of techniques used to rank web pages and discover relevant content on the internet.

Output:


DWM_EXP10.ipynb
File Edit View Insert Runtime Tools Help
Comment
Share
+ Code + Text
Connect
import networkx as nx
import matplotlib.pyplot as plt
def hits_algorithm(graph, max_iter=10, tol=1e-6):
hubs = {node: 1 for node in graph.nodes}
authorities = {node: 1 for node in graph.nodes}
for _ in range(max_iter):
new_hubs = {}
new_authorities = {}
for node in graph.nodes:
new_authority = sum(hubs[neighbor] for neighbor in graph.predecessors(node))
new_authorities[node] = new_authority
for node in graph.nodes:
new_hub = sum(authorities[neighbor] for neighbor in graph.successors(node))
new_hubs[node] = new_hub
norm = max(max(new_hubs.values()), max(new_authorities.values()))
hubs = {node: score / norm for node, score in new_hubs.items()}
authorities = {node: score / norm for node, score in new_authorities.items()}
if all(abs(hubs[node] - new_hubs[node]) < tol for node in graph.nodes) and \
all(abs(authorities[node] - new_authorities[node]) < tol for node in graph.nodes):
break
return hubs, authorities
graph = nx.DiGraph()


DWM_EXP10.ipynb
File Edit View Insert Runtime Tools Help
Comment
Share
+ Code + Text
Connect
graph = nx.DiGraph()
graph.add_edges_from([('A', 'B'), ('B', 'C'), ('B', 'E'), ('C', 'A'),
('D', 'E'), ('E', 'D'), ('E', 'B'), ('E', 'F'),
('E', 'C'), ('F', 'C'), ('F', 'H'), ('G', 'A'),
('G', 'C'), ('H', 'A')])
plt.figure(figsize=(10, 10))
nx.draw_networkx(graph, with_labels=True)
hubs, authorities = hits_algorithm(graph, max_iter=10)
print("Hub Scores: ", hubs)
print("Authority Scores: ", authorities)
Hub Scores: {'B': 0.5235071806500377, 'E': 0.8896447467876035, 'D': 0.10642479213907781, 'G': 0.5668178382464095, 'C': 0.14973544973544972, 'H': 0.14973544973544972, 'A': 0.1768707482
Authority Scores: {'B': 0.436885865457294, 'E': 0.25245653817082386, 'D': 0.363416477702192, 'G': 0.0, 'C': 1.0, 'H': 0.2015117157974300 7, 'A': 0.3392290249433106 4, 'F': 0.3634164777
D
A


DWM_EXP10.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Connect
+ Code
+ Text
Hub Scores: {'B': 0.5235071806500377, 'E': 0.8896447467876035, 'D': 0.10642479211907781, 'G': 0.5668178382464095, 'C': 0.14973544973544972, 'H': 0.14973544973544972, 'A': 0.17682074829931965, 'F': 0.5013605442176868}
Authority Scores: {'B': 0.4368855865457294, 'E': 0.2524565381708238, 'D': 0.36341647702192, 'G': 0.0, 'C': 1.0, 'H': 0.20151171579743007, 'A': 0.3392290249433106, 'F': 0.36341647702192}
D
A
H
B
E
C
G
F