SOCIAL NETWORK ANALYTICS LAB ASSESSMENT

Name: Matam Tejaswini

Reg no: 22MCB0034

analysis of bibliographic graph and visualization of heterogeneous graph

1. Analysis of Bibliographic Graph:

A bibliographic graph represents the relationships between academic papers, authors, and citations. It can be analyzed to gain insights into the structure of the research domain, identify influential papers or authors, detect research trends, and more. Here are some common analysis techniques:

- Network Analysis: Measure centrality metrics like degree centrality, betweenness centrality, and eigenvector centrality to identify key papers/authors. Explore community detection algorithms (e.g., Louvain, Girvan-Newman) to uncover research clusters.

- Citation Analysis: Examine citation patterns to identify highly cited papers, influential authors, and research lineage. Analyze citation networks to understand the flow of ideas within a field.

- Co-authorship Analysis: Analyze co-authorship networks to identify collaboration patterns, prominent research groups, and interdisciplinary connections.

2. Visualization of Heterogeneous Graph:

Heterogeneous graphs represent diverse entities (e.g., authors, papers, organizations) and their relationships (e.g., authorship, citation, affiliation). Visualizing such graphs can aid in understanding complex networks. Here are some approaches:

- Node-Link Diagrams: Represent nodes as entities and links as relationships. Use different node shapes, colors, and sizes to denote entity types or attributes. Employ force-directed layout algorithms (e.g., Fruchterman-Reingold, Kamada-Kawai) to position nodes based on their relationships.

- Matrix-Based Visualization: Create a matrix where rows and columns represent different entity types, and cells depict the presence or strength of relationships between entities. This approach can help reveal patterns and associations between different types of entities.

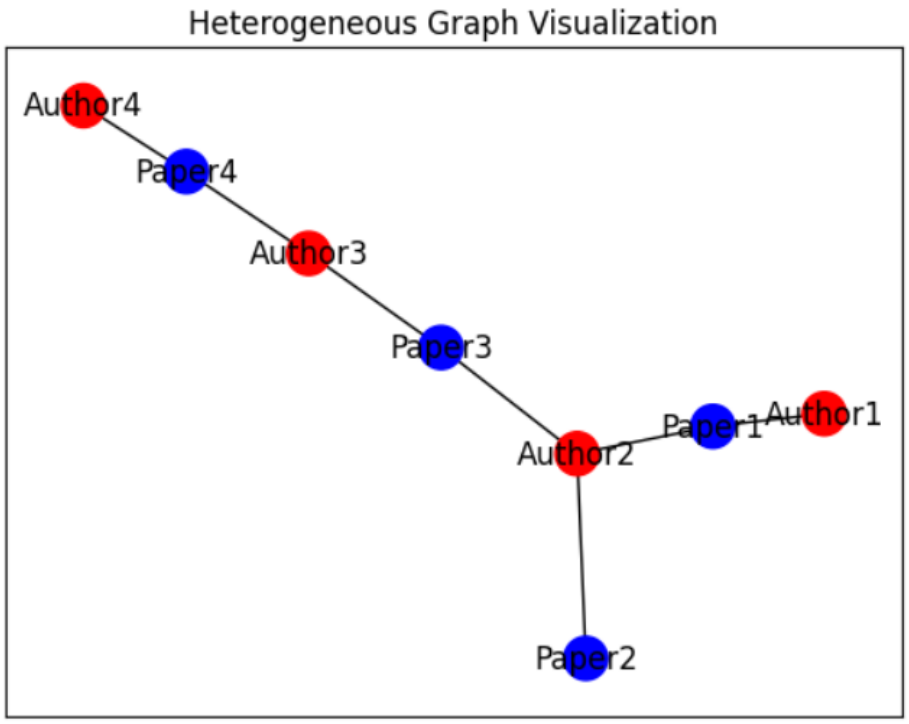
- Interactive Visualization: Develop interactive visualizations that allow users to explore and filter the graph based on specific criteria. Enable zooming, panning, and searching functionalities to facilitate navigation and analysis.

To perform these tasks, we can utilize graph analysis libraries like NetworkX, Gephi, or Neo4j, which offer functionalities for analyzing and visualizing graphs. Additionally, programming languages like Python or R are commonly used for these tasks.

CODE:

```
import networkx as nx
import matplotlib.pyplot as plt
from community.community_louvain import best_partition

# Create a heterogeneous graph
G = nx.Graph()

# Add nodes with different types
G.add_node("Author1", type="author")
G.add_node("Author2", type="author")
G.add_node("Paper1", type="paper")
G.add_node("Paper2", type="paper")
G.add_node("Author3", type="author")
G.add_node("Author4", type="author")
G.add_node("Paper3", type="paper")
G.add_node("Paper4", type="paper")

# Add edges between nodes
G.add_edge("Author1", "Paper1", relationship="authorship")
G.add_edge("Author2", "Paper1", relationship="authorship")
G.add_edge("Author2", "Paper2", relationship="authorship")
G.add_edge("Author2", "Paper3", relationship="authorship")
G.add_edge("Author3", "Paper3", relationship="authorship")
G.add_edge("Author4", "Paper4", relationship="authorship")
G.add_edge("Author3", "Paper4", relationship="authorship")

# Visualize the graph
pos = nx.spring_layout(G)
# Positions the nodes using a spring layout algorithm
node_colors = {'author': 'red', 'paper': 'blue'}
node_types = nx.get_node_attributes(G, 'type')
node_colors = [node_colors[node_types[node]] for node in G.nodes()]
nx.draw_networkx(G, pos, node_color=node_colors)
```

```python
plt.title("Heterogeneous Graph Visualization")
plt.show()

# Perform clustering using Louvain algorithm
partition = best_partition(G)
clusters = {}
for node, cluster_id in partition.items():
    if cluster_id not in clusters:
        clusters[cluster_id] = []
    clusters[cluster_id].append(node)

print("Clusters:")
for cluster_id, nodes in clusters.items():
    print(f"Cluster {cluster_id}: {nodes}")
```




```
Clusters:
Cluster 0: ['Author1', 'Author2', 'Paper1', 'Paper2']
Cluster 1: ['Author3', 'Author4', 'Paper3', 'Paper4']
```