**A red and white logo on a black background

AI-generated content may be incorrect.**

**Artificial Intelligence & Data Science**

**Title:** Mutual Friend Recommender Using BFS and Graph Visualization

Group Name: -

1. Rahil Kothari
2. Jitesh Sidhani
3. Harshit Sahani

Professor: - Sudipta Roy

Teacher Assistant: -

1. Priyanka Mishra
2. Nikita Divya

**Index**

|  |  |
| --- | --- |
| 1 | Introduction |
| 2 | Dataset |
| 3 | Methodology |
| 4 | Results and Discussion |
| 5 | Conclusion |

**1. Introduction**

In the modern digital world, social networks thrive on user connectivity and meaningful engagement. Recommending new friends based on existing relationships enhances user interaction and growth within the platform

This project introduces a Python-based mutual friend recommender system that uses the Breadth-First Search (BFS) algorithm to identify potential friends based on mutual connections. Additionally, the system integrates graph visualization using NetworkX and matplotlib, enabling users to intuitively explore their social network.

Leveraging the Breadth-First Search (BFS) algorithm, the system efficiently traverses the social graph to identify users who share mutual friends but have not yet connected directly. By focusing on mutual connections, the recommender prioritizes suggestions that are more likely to be accepted, improving the quality and relevance of recommended friendships.

**2. Dataset**

The data for this project is a simulated mini social graph, represented using an adjacency list in Python.

* Nodes (Users): Nodes represent the users
* Edges (Friendships): Bi-directional connections
* Format: Dictionary (Adjacency List)

**3. Methodology**

BFS-Based Friend Suggestion

The algorithm recommends friends of friends (2nd-degree connections), excluding direct friends and the user. It traverses the network using a modified BFS approach to calculate mutual friend counts.

Steps:

* Start from the selected user.
* Iterate over each direct friend.
* For each direct friend, collect their connections.
* If the connection is not the user or already a direct friend, increment their mutual count.
* Return a dictionary of suggested users with corresponding mutual friend counts.

The simplicity of this BFS-based method allows for quick computation even on moderately large networks, and its results can be easily visualized using graph tools, enhancing user understanding and engagement with their social graph.

**4. Results and Discussion**

Features:

User Input: Prompt to enter a username.

Output:

* Suggested friends with mutual friend counts.
* Visual graph showing current user and recommendations.
* Graphical View: Launches a matplotlib window displaying the network

The recommender system successfully identifies second-degree connections and prioritizes suggestions based on the number of mutual friends. It is effective even in small social graphs and can be easily scaled.

**Observations:**

* Handles small user bases efficiently.
* Provides intuitive, explainable recommendations.
* Visualization aids in understanding network structure.

**Performance:**

* BFS traversal time: For a user with F direct friends, and assuming each friend has on average K connections

Time Complexity=O(F⋅K) which is negligible(constant time for small graphs)

* Visualization renders smoothly via matplotlib.

**5. Conclusion**

FriendNet exemplifies how simple graph traversal techniques can bring meaningful value to social networking applications. Through mutual friend recommendations and intuitive graph visualization, the system fosters better user connectivity and engagement.

**Future Enhancements:**

* Add GUI input using Tkinter
* Scale to large datasets (e.g., CSV import).
* Implement ranking with weights (e.g., interaction frequency).