Report of Network acquaintance Recommendation system

Ideathon DSA/CSL2020

April 29, 2024

1 Introduction

Social networks have become an integral part of our daily lives, connecting individuals and facilitating communication on a global scale. In this context, friend suggestion systems play a crucial role in enhancing user experience by recommending new connections based on mutual interests or connections. In this report, we analyze and explain the implementation of a Friend Suggestion System using both C and Python programming languages. The system allows users to input social connections, analyze the network structure, and recommend new connections to expand their social circles.

2 Methodology

Friend Suggestion System: Enhancing Social Connections

In an increasingly interconnected world, social networks play a pivotal role in our lives. Whether it's reconnecting with old friends, expanding professional circles, or discovering shared interests, these networks shape our experiences. The Friend Suggestion System, implemented using a combination of C and Python, emerges as a powerful tool to enhance these connections.

Approach:

Input Data: The initial step involves gathering input data regarding the total number of nodes (users) in the graph and constructing an adjacency matrix to represent the connections between nodes. The adjacency matrix indicates whether a link exists between two nodes, signifying a relationship or association.

Identifying Adjacent Nodes: Next, the system computes the count of adjacent nodes for each node in the graph. This step involves traversing the adjacency matrix and determining the number of direct connections (edges) associated with each node.

Recommendation Generation: The recommendation generation process entails analyzing the connections of adjacent nodes to identify potential friends

Friend suggestion

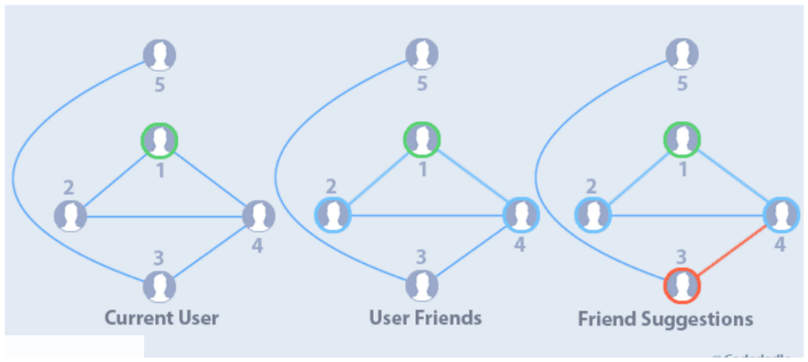


Figure 1: Graphs Application

or connections for a given user. This involves traversing the graph and considering the connections of each adjacent node to suggest new connections for the user.

Stack Implementation: To efficiently manage and track recommended friends, a stack data structure is employed. The stack stores the recommended friends for the user, ensuring that each recommendation is unique and avoiding duplication.

Friend Request Handling: Upon generating friend recommendations, the system prompts the user to decide whether to send friend requests to the recommended individuals. If the user accepts a recommendation and sends a friend request, the corresponding link is added to the adjacency matrix, reflecting the establishment of a new connection between users.

Understanding the System **Graph Representation:** At its core, the system models social relationships as a graph. Each user corresponds to a node, and friendships are represented by edges connecting these nodes. Adjacency matrices efficiently capture these connections. A binary entry (1 or 0) in the matrix signifies whether two users are friends or not.

Graph Algorithms: Leveraging graph algorithms, the system explores the network's structure. Depth-First Search (DFS) and Breadth-First Search (BFS) traverse the graph, revealing hidden connections. Community detection algorithms identify clusters of closely connected users, unveiling subgroups within the larger network.

Recommendation Engine: The heart of the system lies in its recommendation engine. By analyzing existing friendships, it suggests potential new connections. Collaborative filtering techniques, content-based filtering, and hybrid approaches personalize recommendations based on user preferences, mutual friends, and shared interests.

Adjacency Matrices:

These matrices succinctly represent the social graph. Rows and columns correspond to users, and entries indicate friendship status. Matrix operations facilitate efficient computations, such as finding common friends or predicting missing edges.

Graph Visualization Techniques

Node Positioning: Force-directed layouts position nodes based on attractive and repulsive forces. Users with stronger connections gravitate toward each other.

Circular layouts group friends together, emphasizing clusters.

Edge Styling: Edge thickness or color intensity reflects the strength of friendship. Closer friends have thicker or darker edges. Dotted lines represent potential connections, encouraging users to explore new relationships.

Community Visualization: Highlighting communities fosters a sense of belonging. Color-coded clusters reveal shared interests or geographic proximity. Overlaying geographical maps provides context—connecting users from the same city or neighborhood. Impact and User Experience

Discovering Hidden Ties: The system uncovers latent connections. Users may find childhood friends, distant relatives, or colleagues they didn't know were on the platform. Serendipitous encounters enrich the user experience.

Meaningful Recommendations: Beyond suggesting mutual friends, the system recommends users with complementary interests. Book lovers connect over favorite authors; hobbyists find fellow enthusiasts. These meaningful connections enhance user satisfaction.

Privacy Considerations: Balancing recommendations with privacy is crucial. Users control visibility and can choose to accept or decline suggestions. Anonymized data ensures privacy while optimizing recommendations. Future Directions

Dynamic Networks: Real-world social networks evolve. The system adapts to changing friendships, incorporating temporal dynamics.

Incorporating Context: User profiles, posts, and interactions provide context. Natural language processing (NLP) analyzes textual data for better recommendations.

Ethical Friend Suggestions: Ensuring diversity and avoiding echo chambers is essential. The system promotes connections across diverse backgrounds

3 C Code Analysis

3.1 Overview

The C code implements the core functionalities of the system, including adjacency matrix handling, friend recommendation, and CSV export.

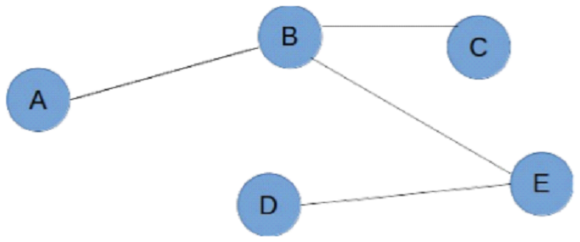


Figure 2: Data structure

3.2 Functions

- `adjacency_count`: Counts the number of adjacent nodes for a given node in the adjacency matrix.
- `adjacency_result`: Finds adjacent nodes for a given node and stores them in an array.
- `write_to_csv`: Writes the adjacency matrix to a CSV file.
- `main`: Main function to interact with the user and perform operations based on user input.

3.3 Data Structures

- `matrix`: 2D array representing the adjacency matrix.
- `stack`: Array used for stack operations.

3.4 Usage

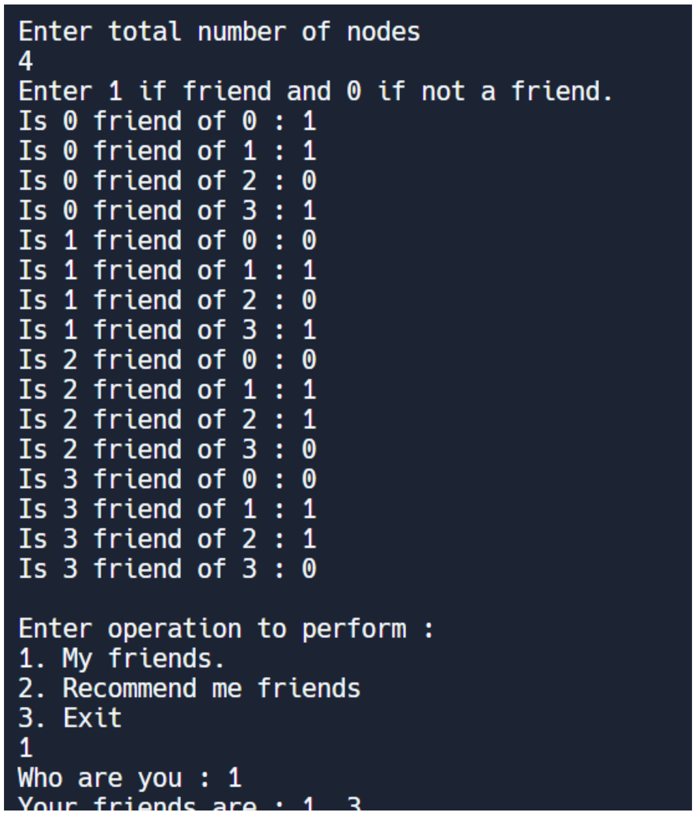
The user can input social connections, choose operations such as displaying friends or recommending new connections, and exit the program.

3.5 Overview

The Python code visualizes the social network graph using NetworkX and Matplotlib based on the adjacency matrix generated by the C code.

3.6 Libraries

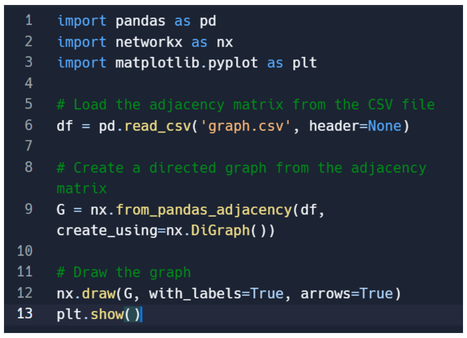
- `pandas`: For reading the CSV file containing the adjacency matrix.
- `networkx`: For creating and manipulating the graph structure.
- `matplotlib`: For graph visualization.



```
Enter total number of nodes
4
Enter 1 if friend and 0 if not a friend.
Is 0 friend of 0 : 1
Is 0 friend of 1 : 1
Is 0 friend of 2 : 0
Is 0 friend of 3 : 1
Is 1 friend of 0 : 0
Is 1 friend of 1 : 1
Is 1 friend of 2 : 0
Is 1 friend of 3 : 1
Is 2 friend of 0 : 0
Is 2 friend of 1 : 1
Is 2 friend of 2 : 1
Is 2 friend of 3 : 0
Is 3 friend of 0 : 0
Is 3 friend of 1 : 1
Is 3 friend of 2 : 1
Is 3 friend of 3 : 0

Enter operation to perform :
1. My friends.
2. Recommend me friends
3. Exit
1
Who are you : 1
Your friends are : 1  3
```

Figure 3: Sample Output



```python
import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt

# Load the adjacency matrix from the CSV file
df = pd.read_csv('graph.csv', header=None)

# Create a directed graph from the adjacency
matrix
G = nx.from_pandas_adjacency(df,
create_using=nx.DiGraph())

# Draw the graph
nx.draw(G, with_labels=True, arrows=True)
plt.show()
```

Figure 4: Python

4 Python Code Analysis

4.1 Functionality

- Loads the adjacency matrix from the CSV file.
- Creates a directed graph using NetworkX.
- Visualizes the graph with node labels and directed edges using Matplotlib.

5 Conclusion

The Friend Suggestion System implemented in C and Python provides a comprehensive solution for analyzing social connections and recommending new connections. By leveraging adjacency matrices and graph visualization techniques, the system offers insights into social networks, ultimately enhancing user experience and fostering meaningful connections.

A complete solution for examining social connections and suggesting new ones is offered by the C and Python-based Friend Suggestion System. The technology provides insights into social networks by utilizing adjacency matrices and graph visualization techniques, which ultimately improves user experience and fosters meaningful interactions.

In conclusion, the utilization of graph structures offers an effective approach to address the friend suggestion or recommendation problem in social media platforms. By leveraging the relationships and associations encoded in the graph, the system can provide personalized recommendations to users, facilitating the expansion of their social networks.

Through the systematic analysis of connections and the implementation of efficient data structures and algorithms, the friend recommendation system enhances user engagement and fosters meaningful connections within the social network.

This approach underscores the significance of graph theory and data-driven solutions in addressing complex problems in social media platforms, paving the way for enhanced user experiences and network dynamics.

As social networks continue to evolve, leveraging graph-based approaches remains integral to optimizing friend recommendation systems and driving user interaction and connectivity.