Social Media Friend Suggetion System

Using DSA

DSA BOOTCAMP -2024

Computer Engineering & Information Technology

Submitted By:

Sanjeet Kumar (2247349)

Sahil Raja (2248419)

Sanish Kumar (2246004)

Saloni Kumari(2247721)

Sandeep Kr Gupta (2247246)

Submitted To:

Deepti Ma'am

(Bootcamp Trainer)

highlights

➢ Introduction

The Social Media Friend Suggestion System uses graph theory and the BFS algorithm to suggest new friends based on mutual connections.

➢ Objectives

Build a system to suggest friends based on existing connections.
Use graph traversal to find potential second-degree friends.
Demonstrate practical use of Breadth-First Search (BFS) in solving real-world problems.

➢ Tools & Language Use

Python: Primary programming language.
Libraries: collections (for deque and defaultdict).

➢ Technology

Graph Theory: Model users as nodes and friendships as edges.
Breadth-First Search (BFS): For finding friend suggestions by exploring second-degree connections.
Data Structures: Graphs, sets, queues (deque).

Code Overview

```
from collections import deque, defaultdict
# Social media graph where users are nodes and
friendships are edges
class SocialMediaGraph:
    def __init__(self):
        # Use defaultdict to automatically create lists
 for each user node
        self.graph = defaultdict(list)

    # Add a friendship (edge) between two users
    def add_friendship(self, user1, user2):
        self.graph[user1].append(user2)
        self.graph[user2].append(user1)  # Assuming
undirected graph (mutual friendship)
# Breadth-First Search for friend suggestion
    def friend_suggestions(self, user):
        visited = set()  # Track visited nodes
        queue = deque([user])  # Queue for BFS
starting from the given user
        suggestions = set()  # Store friend suggestions
# Mark the user and their direct friends as visited
        visited.add(user)
        # Get the direct friends (first-degree
connections) of the user
        direct_friends = set(self.graph[user])
        visited.update(direct_friends)
# Debugging output
        print(f"User -> {user}, Direct Friends -> {direct_friends}")

        # BFS: Find second-degree friends
        for friend in direct_friends:
            # Explore friends of friends
            for friend_of_friend in self.graph[friend]:
                if friend_of_friend not in visited:
                    suggestions.add(friend_of_friend)
        return suggestions
# Testing the BFS-based friend suggestion system
if __name__ == "__main__":
    # Create a new social media graph
    smg = SocialMediaGraph()
    # Add friendships
    smg.add_friendship("Alice", "Bob")
    smg.add_friendship("Alice", "Charlie")
    smg.add_friendship("Bob", "David")
    smg.add_friendship("Charlie", "Emma")
    smg.add_friendship("David", "Frank")
    smg.add_friendship("Emma", "George")
    # Debugging: print the graph structure
    print("Graph connections:", dict(smg.graph))
    # Example: Find friend suggestions for Alice
    suggestions = smg.friend_suggestions("Alice")
    print("Friend suggestions for Alice:", suggestions)
```

➢ Industrial Application & Purpose

Applications:

Social Media platforms (Facebook, LinkedIn) for user retention.

Recommendation systems to increase user interaction.

Purpose: Enhance user engagement by suggesting meaningful connections.

➢ Future Expansions

Advanced Recommendations: Incorporating machine learning to improve suggestion accuracy.

Scalability: Optimizing graph algorithms for larger user bases.

Integration with AI: Predicting connections based on user behavior.

➢ Conclusion

This system demonstrates how BFS and graph theory can be applied in real-world social networks.

It provides a scalable solution for suggesting new friendships based on mutual connections.

Future work could involve AI and machine learning for more personalized recommendations.

Thank you