Six Degrees of Separation in the Facebook Social Network

Introduction

This project aims to explore the "Six Degrees of Separation" concept in the Facebook social network. The goal is to determine the shortest path between two users in the network, using the dataset provided by the Stanford Large Network Dataset Collection.

Dataset

The dataset used in this project is the "Social circles: Facebook" dataset. It consists of the following files:

- `facebook\_combined.txt`: Edges from all egonets combined.

- `readme-Ego.txt`: Description of files.

For this project, I primarily use the `facebook\_combined.txt` file, which contains the edges of the entire network.

Dataset Statistics

- Nodes: 4039

- Edges: 88234

- Average Clustering Coefficient: 0.6055

- Number of Triangles: 1612010

- Diameter (longest shortest path): 8

- 90-percentile Effective Diameter: 4.7

Implementation

The project is implemented in Rust using the petgraph library for graph operations. The main steps are:

1. Load and Parse the Dataset

- The data\_loader.rs module reads the facebook\_combined.txt file and constructs an undirected graph.

- The graph is represented using petgraph::graph::UnGraph.

2. Find the Shortest Path

- The main.rs file uses Dijkstra's algorithm to find the shortest path between two specified nodes.

- The start and target nodes are specified as command-line arguments or hardcoded in the main function.

3. Visualize the Network and Shortest Paths

- The visualization.rs module provides a way to visualize the network and highlight the shortest path.

- The visualization is done using the plotters library.

- The visualization process can be time-consuming due to the complexity of the graph and the rendering tasks.

4. Testing

- The tests.rs module contains unit tests to ensure the correctness of the implementation.

- The tests include verifying the shortest path lengths and the structure of the graph.

Code Structure

- main.rs: Entry point of the application. Loads the graph, finds the shortest path, and visualizes the result.

- data\_loader.rs: Contains the logic to read the dataset and construct the graph.

- visualization.rs: Handles the visualization of the graph and the shortest path.

- tests.rs: Contains unit tests to verify the correctness of the implementation.

The output will display the shortest path length between the specified nodes and generate a visualization of the graph with the shortest path highlighted.

Current Challenges

- Performance: The visualization process can be slow, especially for large graphs. This is due to the computational intensity of layout calculations and rendering.

Conclusion

This project demonstrates the application of graph theory and algorithms to real-world social networks. By finding the shortest path between two users in the Facebook social network, we can gain insights into the connectivity and structure of the network. The project also highlights the challenges and potential areas for improvement in handling and visualizing large graph datasets.