Jaccard Similarity Analysis of the Facebook Combined Network Dataset Using Rust

Project Overview

My Rust project is designed to delve into the Facebook combined network dataset from the Stanford Network Analysis Project (SNAP), focusing specifically on understanding social connectivity and community structures. (https://snap.stanford.edu/data/ego-Facebook.html)

The primary goal of this project is to apply Jaccard similarity measures to identify and analyze the relationships between users within the network. By investigating how users are interconnected, particularly through mutual friends, this project aims to uncover influential nodes and clusters that significantly impact the flow of information.

The Facebook dataset provides a comprehensive representation of social interactions, where nodes correspond to individual users and edges signify mutual friendships. This setup creates a complex web of relationships, making it an ideal model for studying social network behaviors and interaction patterns.

In this analysis, I utilize Rust's efficient processing capabilities to handle the dataset, employing the petgraph library to model the network as an undirected graph. This approach reflects the reciprocal nature of Facebook friendships. It also aligns with my interest in exploring how mutual connections influence social group formations and community ties.

Technical Approach

Utilizing Rust's powerful data manipulation capabilities, the project is structured around several core modules:

1. main.rs

- Purpose and Functionality:
    - Acts as the orchestrator for integrating the functionalities provided by other modules, ensuring streamlined execution from data loading to processing and visualization.
    - Coordinates the entire program workflow, starting with data loading, invoking analysis functions, and managing output processes like data visualization.

2. network_graph.rs

- Purpose and Functionality:
    - Manages the graph's data structure, facilitating efficient manipulations and queries.

- Key Functions:
    - new(): Initializes a new, empty graph structure.
    - add_node(): Adds a node to the graph if it doesn't already exist.
    - connect_nodes(): Adds an edge between two nodes, creating a direct connection in the graph.
- These functions form the backbone for managing the graph structure, essential for subsequent analytical computations.

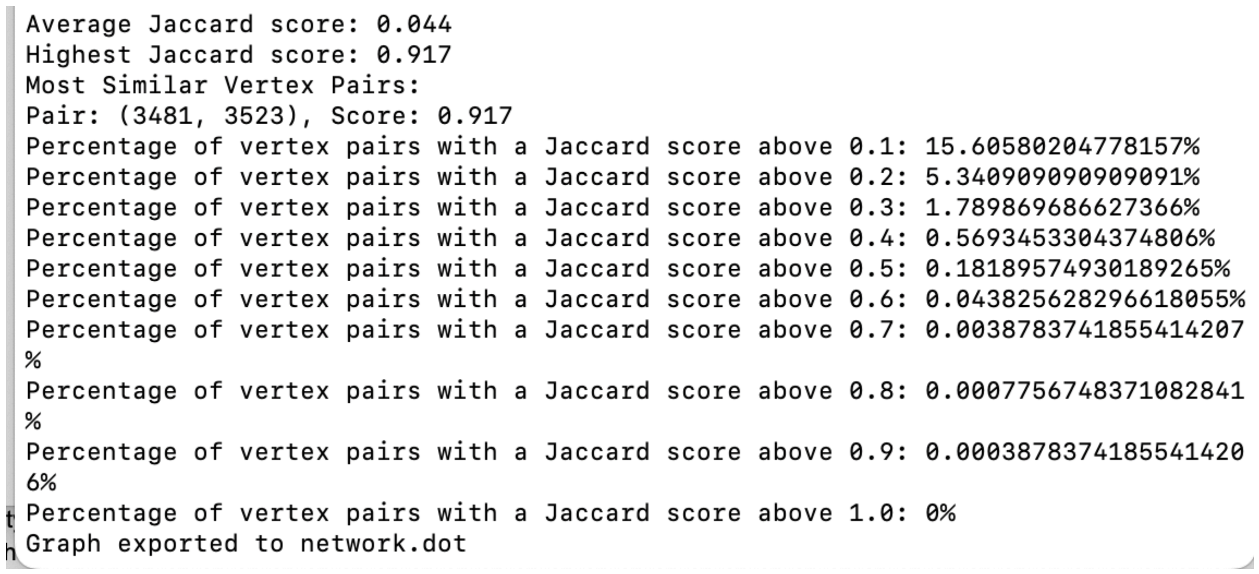
3. similarity_analysis.rs

- Purpose and Functionality:
    - Dedicated to computing Jaccard similarities, focusing on discerning node pairs that exhibit direct and second-level connections.
- Key Functions:
    - calculate_jaccard_scores(): Computes Jaccard similarity scores for pairs of nodes, particularly those that are two steps apart.
    - summarize_scores(): Summarize the calculated Jaccard scores into average, maximum, and list of most similar pairs.
    - compute_thresholds(): Compute the percentage of node pairs exceeding specific Jaccard similarity thresholds.
- Crucial for uncovering potential community ties and understanding the network's social clusters.

4. graph_export.rs

- Purpose and Functionality:
    - Supports the visualization process by exporting the network data to DOT format, enabling graphical representation using Graphviz.
- Key Function:
    - export_to_dot(): Converts the internal graph data into DOT format, which can be visualized with graph visualization tools to provide a graphical representation of the network structure.
    - Essential for visualizing complex network relationships and providing insights that are easier to interpret visually.

Results and Analysis



```
Average Jaccard score: 0.044
Highest Jaccard score: 0.917
Most Similar Vertex Pairs:
Pair: (3481, 3523), Score: 0.917
Percentage of vertex pairs with a Jaccard score above 0.1: 15.60580204778157%
Percentage of vertex pairs with a Jaccard score above 0.2: 5.340909090909091%
Percentage of vertex pairs with a Jaccard score above 0.3: 1.789869686627366%
Percentage of vertex pairs with a Jaccard score above 0.4: 0.5693453304374806%
Percentage of vertex pairs with a Jaccard score above 0.5: 0.18189574930189265%
Percentage of vertex pairs with a Jaccard score above 0.6: 0.043825628296618055%
Percentage of vertex pairs with a Jaccard score above 0.7: 0.0038783741855414207
%
Percentage of vertex pairs with a Jaccard score above 0.8: 0.0007756748371082841
%
Percentage of vertex pairs with a Jaccard score above 0.9: 0.0003878374185541420
6%
Percentage of vertex pairs with a Jaccard score above 1.0: 0%
Graph exported to network.dot
```

Upon running the application, it outputs relevant information as follows:

Basic Graph Information:

1. Number of nodes: 4039
2. Number of edges: 88234

Jaccard Similarity Scores:

1. Mean Jaccard similarity: 0.044
2. Max Jaccard similarity: 0.917

Pairs with the Highest Similarity:

1. Pair: (3481, 3523), Similarity: 0.917

Percentage of Pairs with Jaccard Similarity Above Thresholds:

1. Above 0.1: 15.61%
2. Above 0.5: 0.18%
3. Above 0.9: 0.00039%

- **Average Jaccard Score**: Calculated at 0.044, suggesting a low level of immediate node-to-node similarity across the network, which is typical in diverse and large-scale social networks.
- **Highest Jaccard Score:** Notably, a maximum score of 0.917 between nodes (3481, 3523) indicates a highly interconnected pair, likely sharing a significant mutual community.

- **Community Detection:** By analyzing nodes with Jaccard scores above certain thresholds, the project identifies clusters of users potentially forming tight-knit communities, crucial for targeted network interventions.

Visualization and Documentation

A segment of the network has been visualized and exported to a DOT file, now hosted on our GitHub repository. This visualization highlights areas of dense connectivity and significant nodes, offering a graphical interpretation of the network's structure.

Testing

In this Rust project, comprehensive testing is carried out to ensure the robustness and accuracy of the network analysis functionalities. Tests include verifying the correct loading of the graph from data files, ensuring the Jaccard similarity scores are calculated correctly between node pairs, and validating that centrality measures accurately reflect the network's properties.

Future Enhancements

- Temporal Dynamics: Introducing temporal data analysis could provide insights into the evolution of the network over time, revealing how social ties develop or decay.
- Advanced Graph Algorithms: Implementing more sophisticated algorithms for community detection and network flow could uncover deeper insights into the dynamics of the Facebook social graph.

Conclusion

This project not only enhances our understanding of social structures within large networks but also showcases the efficiency of Rust in handling complex data analysis tasks. The calculated Jaccard similarities contribute to our understanding of social connectivity, influencing strategies in information dissemination, and community management within digital social platforms.