Project Title:

Analysis of Movie Interconnectivity in the MovieLens Dataset

Objective:

To investigate the structural and relational aspects of movies within the MovieLens dataset through user ratings and tags, focusing on understanding the patterns of connectivity that influence movie recommendations.

Data Set Description:

This dataset (ml-latest-small) encompasses 100,836 ratings and 3,683 tag applications across 9,742 movies, generated by 610 users from March 29, 1996, to September 24, 2018. The rich user interaction data makes it an excellent candidate for analyzing movie relationships through graph theory techniques.

Explanation of data.rs:

Structs: Each struct (Movie, Rating, Tag) corresponds to a record in your CSV files. Using #[derive(Deserialize)] allows the csv crate to automatically map CSV columns to struct fields.

Loading Functions: Each function (load\_movies, load\_ratings, load\_tags) reads from a file path, parses the CSV, and deserializes it into a vector of the appropriate struct. These functions return a Result, which is either Ok containing the vector of records or an Err with an error message.

Result:

Movies: Contains information about movies, identified by movieId, along with their title and genres.

Ratings: Includes user ratings for movies, with userId, movieId, rating, and the timestamp of the rating.

Tags: Contains tags that users have applied to movies, described by userId, movieId, tag, and timestamp.

Links: Offers external identifiers for movies, linking movieId to imdbId and tmdbId.

Proposed Analyses:

Graph Construction:

Nodes: Each node represents a movie.

Edges: Connect movies if they share significant user rating patterns or tags.

**Explanation:**

Nodes and Edges: The nodes HashMap stores movie titles indexed by their movie IDs. The edges HashMap uses a set for each movie to store connections (i.e., edges) to other movies, ensuring that each connection is unique and bidirectional.

Constructors and Methods:

new(): Initializes an empty graph.

add\_node(): Inserts a new node if it does not already exist.

add\_edge(): Adds a connection between two nodes, initializing their adjacency list if they don't already exist.

contains\_node(): Checks if a node exists in the graph.

neighbors(): Retrieves a reference to the set of neighbors (connected nodes) of a given node.

num\_nodes() and num\_edges(): Provide quick ways to get the count of nodes and edges, respectively.

Six Degrees of Separation:

Goal: Assess the average path length between movies and examine the 'six degrees of separation' concept within the context of the MovieLens network.

Method: Utilize Breadth-First Search (BFS) and Shortest Paths algorithms to compute distances.

**Explanation:**

Shortest Paths: The compute\_shortest\_paths function uses Breadth-First Search (BFS) to calculate the shortest paths from a starting node to all other reachable nodes in the graph, recording the distances in a hashmap.

Graph Clustering and Partitioning:

Goal: Cluster the network to identify representative movies or genres that define the movie landscape.

Method: Apply graph clustering techniques to partition the network and evaluate the effectiveness of these clusters against intuitive genre classifications.

**Explanation:**

Cluster Identification: The identify\_clusters function (along with its helper, bfs\_cluster) performs a basic form of community detection by using BFS to find connected components of the graph.

Centrality Measures:

Goal: Identify key influential movies based on their centrality within the graph.

Method: Calculate centrality metrics like closeness, betweenness, and degree centrality to pinpoint central movies.

**Explanation:**

Centrality Measures: The compute\_centrality function computes the closeness centrality for each node. It is based on the reciprocal of the sum of the shortest path distances from the node to all other nodes.

Testing Strategy:

Unit Testing: Each component of the graph construction and analysis algorithms will undergo thorough unit testing to verify functionality.

**Explanation of Tests:**

Graph Operations: The test\_add\_nodes\_and\_edges ensures that nodes and edges are added correctly and that their counts are accurate.

Shortest Paths: The test\_compute\_shortest\_paths verifies that the shortest path distances are computed correctly for a simple path scenario.

Centrality Measures: The test\_compute\_centrality checks the correctness of centrality calculations, ensuring that the nodes have expected centrality values based on their positions in the graph.

Cluster Identification: The test\_identify\_clusters confirms that the clustering function correctly identifies all nodes as part of a single cluster in a fully connected component.

Integration Testing: Systems tests will ensure seamless integration between the data preprocessing, graph construction, and analysis modules.

Validation Testing: The outcomes of the graph metrics will be validated against theoretical expectations and existing literature on similar movie networks.

Conclusion:

The project aims to leverage graph theory to uncover underlying patterns of movie recommendations within the MovieLens database, offering insights into how movies are interconnected through user behaviors. This analysis will not only enhance understanding of movie relationships but also potentially improve recommendation systems by identifying influential movies and defining clusters within the network.

Result and Analysis:

Node 2712's connectivity to its direct neighbors (nodes 2016, 3252, 107, 5108, 91353, 78088, and others) reflects its primary interaction layer within the network. These direct connections are crucial as they represent the node’s strongest and most immediate links. Being directly connected to multiple nodes, 2712 potentially acts as a pivotal connector or hub within its network segment, directly impacting its nearest nodes.

The nodes that are two steps away (such as 163937, 172497, 166534, 114554, and others) illustrate a secondary tier of influence and connectivity. These nodes, while not directly connected to 2712, are influenced through its immediate neighbors. This layer is significant for understanding how node 2712 indirectly facilitates broader interactions within the network, extending its reach beyond its immediate neighbors and potentially serving as a bridge in the flow of network traffic.

Implications for Network Analysis：

Community Detection: Analyzing how node 2712 interacts with these two layers of neighbors could be pivotal in identifying network communities or clusters. Algorithms like Louvain or Girvan-Newman could be particularly useful in discerning whether node 2712 is part of a tightly-knit community or if it serves as a gateway between multiple communities. This can reveal insights into the structural cohesion of the network and the role of node 2712 as either a peripheral or a core member of its community.

Centrality Measures: Exploring centrality measures for node 2712 and its connected nodes can further elucidate their roles within the network. Betweenness centrality would highlight nodes that serve as important bridges or points of control within the network, indicating nodes that frequently lie on the shortest paths between other nodes. Closeness centrality, on the other hand, would reveal how quickly information from node 2712 can reach all other nodes in the network, a useful measure of its strategic positioning. Eigenvector centrality could also be useful to determine the influence of node 2712 based on the centrality of its connections, giving weight to nodes that are themselves well-connected.

Path Analysis: Deeper investigation into path lengths and their distribution can offer insights into the network's efficiency in information or resource flow. This includes identifying potential bottlenecks where over-reliance on particular nodes could pose risks to network robustness or finding super-connectors that significantly enhance connectivity and flow within the network.

By building on this foundational analysis of node 2712, stakeholders can better understand the structural characteristics of the network, which is critical for optimizing network flows, understanding social dynamics, or designing more robust and efficient networks. Such insights are not only academically interesting but have practical implications for network design and strategy in various fields, from telecommunications to social media, and beyond.

Reflection:

In this project, I explored the interconnectivity within the MovieLens dataset to uncover relationships and patterns among movies based on user ratings. Utilizing Rust programming, the primary objective was to build an efficient system to analyze a complex network of movie connections. However, a notable shortcoming was the long processing times experienced during the analysis phase. The complexity of the graph, combined with the large volume of data, significantly impacted performance. While Rust's concurrency features were leveraged to improve efficiency, the time required to process connections and calculate centrality metrics was still considerable. Additionally, the project could benefit from further optimization, particularly in how data is loaded and handled during computation. Despite these challenges, the project offered valuable insights into the potential of network analysis in understanding user preferences and movie relationships, highlighting areas for future refinement and the need for more efficient data handling techniques.

Dataset Link:

<https://grouplens.org/datasets/movielens/latest/>