Explanations for the Program

Through this Rust code, it builds a graph-based model that analyzes and forms relationships and connections between the actors and the corresponding movies. Each vertex or node represents a movie, while each edge is formed, it displays two movies that at least one actor partakes in. The program ultimately calculates and outputs the similarity and dissimilarity values between each of the movies in the dataset based on the number of shared actors.

Main Function (main.rs)

- It performs several key operations to manage and analyze the movie graph based on the data from the CSV file. By using the 'load_graph_from_csv' function, the code is able to load a graph structure from the csv file. It reads the movie titles and associated actors and returns a 'Graph' object where each movie becomes a node and actors are categorized and recorded as sets corresponding to their movies. After loading the graph, it prints the total number of movies (nodes) in the graph. The 'main' function calls on the 'get_most_similar' on the 'Graph' object to determine the two movies with the highest Jaccard similarity score, which measures the similarity between the sets of actors for two movies; it is defined as the size of the intersection divided by the size of the union of the actors sets. Ultimately, if a pair is found after traversing the graph, it prints out the titles of the following movies with their similarity score. As for the 'get_most_dissimilar' function, it is called to find the two movies with the highest Jaccard dissimilarity score, where the Jaccard dissimilarity is usually 1 subtracted by the Jaccard similarity, showing how different the movies are in terms of their actors. If a pair is found, it prints out the titles of the following movies with their dissimilarity score. As for the outputs, it would show 1 for the similarity output, indicating that it is completely similar, and it would also show 1 for the dissimilarity output, showing that it is completely dissimilar. If no such movies are found after calculating the similarity and dissimilarity, it prints out a message that none for each potential output is found.

Extra Function 1 (file_io.rs)

- Defines a function 'load_graph_from_csv' which is used to read the movie data from the csv file and populate the graph structure with it. First it imports all the necessary module references to handle file operations and input/output operations. The function 'load_graph_from_csv' takes a '&str' path to a csv file and returns a 'Graph' object. It opens and reads the file while performing error handling. Line processing is also done with the function iterating over each line of the file while each of them is checked to see if it is able to read without errors, which is then split by commas into parts using 'split(',')'. Eventually, it extracts and processes data, ensuring there is enough data to work with by checking the 'parts' vector. The movie titles are extracted and trimmed of any whitespace that could exist and the strings are split into individual actor names and then converted to a 'String' type before being stored into a vector. The 'add_movie'

method ultimately is called with the movie and title and the list of actors, which is responsible for adding them as the nodes and edges in the graph. The ‘Graph’ object is then returned.

Extra Function 2 (similarity.rs)

- This part defines two functions to compute the Jaccard similarity and dissimilarity between two sets. The ‘jaccard_similarity’ calculates the similarity between two sets, which then measures how similar two sets are and is defined as the size of the intersection divided by the size of the union of the sets. It calculates the intersection of ‘set1’ and ‘set2’ which are the elements common to both sets. As for the dissimilarity, it is complementary to the Jaccard similarity, and uses the same parameters. It computes it by subtracting the Jaccard similarity from 1. This value shows the proportion of elements that are not shared between two sets relative to the total number of unique elements across both sets.

Results

To run this program, you must first download this project to a local software then initiate the ‘cargo build’ command to actually build the project. The next command to use is ‘cargo run –release’ which is a command that is mostly recommended to use when wanting for faster results. Another command ‘cargo run’ could be used, but would take a much longer time to see the outputs.

A sample output given:

- Graph loaded with 8795 movies.
  The most similar movies are 'Power Rangers Samurai: Party Monsters (Halloween Special)' and 'Mumbai Delhi Mumbai' with a similarity score of 1.
  The most dissimilar movies are 'Power Rangers Samurai: Party Monsters (Halloween Special)' and 'Country Ever After' with a dissimilarity score of 1.

The output is what I wanted where both displays two movies with the highest similarity and dissimilarity values (1). We can see how according to the similarity, it may show common actors that frequently played in those film genres, while for the dissimilarity displays that does not have any similarities for actors that participated in those films.

Other

Why This Project

- This project was completed to experience using graph data structures to analyze the relationships between actors and the movies. Generally, graph data structure is a powerful tool to create and display relationships between multiple things. Through this project, I was able to apply the techniques in graph analysis to a real-world problem or data; it

further gave me insight and knowledge about the graph data structures and algorithms by deeply exploring their concepts and applying it to certain scenarios.

Dataset (csv file)

- Despite not being my first choice, I chose this data set since it aligned well with what I wanted to find or learn more about. It contained the actors and the corresponding movie names, which allowed me to implement this project idea. I found this dataset on Kaggle that provided information regarding netflix shows, showing actors, shows/movies, years, countries, etc.
- https://www.kaggle.com/datasets/shivamb/netflix-shows

Challenges

Unfortunately, there were problems in linking the github repository to the VSCode in the beginning, but still took a very long time to figure it out. There was problem with the designated branches between VSCode and github. However I was able to tackle it by doing it on Jupyter instead. It does not show the accurate time lapse unfortunately.