IMDb Movies

I have a deep passion for film and people's differing opinions on cinema. For my project, I wanted to analyze trends in IMDb movie data using different graph theory as I focused on genre similarities and rating patterns. As I snooped around with different datasets, I found it interesting to see how different audiences evaluate cinema and see the discrepancy between the two scores of critic scores and user scores. There is hefty information on dates and genres that gives space to examine trends in the differences int the scores over time and across genres. Because of how much curiosity the data set sparked in me, I really wanted to investigate into how movies relate based on their metadata and ratings, identify influential films, and understand genre-based clustering and historical trends. To narrow in on a specific question, I would like to guide my project to answer: how does the reception from critics and the public with commercial success change over time and across different genres of film?

Here is the data set I ended up using for my project:

https://www.kaggle.com/datasets/samruddhim/imdb-movies-analysis

Size:

To kick off the investigation, I started with data processing. The CSV file is loaded using the csv crate and serde. The missing entries were filtered out. Removed rows that missed the critic or user scores. Then the Metacritic scores were normalized to fit a 0 to 10 scale. A score gap was computed that took the IMDb rating and subtracted it by the

Metacritic score. Then parsed year from string to numereic decade format. Genre was split into a vector of genre tags.

For the different modules I had created, I had data_cleaning.rs to load and clean the dataset, graph_builder.rs to construct similarity graph based on genre and score, analysis.rs for centrality, clustering, and a score gap analysis, main.rs to carry out everything and write the output files.

Looking at the key structs and functions, there are a few of them that should be highlighted. The movie struct has the raw movie data from CSV with the fields being title, year, genre, metacritic, IMDb ratings. The CleanMovie struct is a cleaned and parsed version of this essentially but adds parsed year, a genre vector, the normalized scores, and score_gap. The MovieNode is a graph node containing only essential information to compare the films.

Then moving to the functions, build_graph() inputs the slice of CleanMovie, genre weight, score weight, and similarity threshold, uses Jaccard similarity and cosine similarity to link similar films. I used this source:

https://users.rust-lang.org/t/jaccard-similarity-in-rust-polars/93451

as a way to understand jaccard similarity logic. Then the function will output a undirected graph with movies as the nodes. degree_centrality() counts the number of direct connections or edges each movie node has, ultimately showcasing how inherently popular or central movie is in the graph. betweenness_centrality() will approximate the amount of times a movie will fall along the shortest path between others which is based on this source:

https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/

that I found helpful to understand more on Dijkstra's algorithm to finding the shortest path. average_score_gap_by_decade() will group movies by decade and averages score gap, while average_score_gap_by_genre() will group movies by genre and averages score gap. The last function I want to highlight is the find_clusters() that uses Breadth-First Search in order to find the connected components in the graph. Each cluster will represent a group of similar movies.

The main workflow loads and clean data through Vec<CleanMovies>, creates graph using build_graph(), computes the centrality and trends metrics, writes results to CSV and TXT files for output, and then I also have a python script that I created on Google Colab to visualize the results.

For running tests, when using cargo test in the terminal, I will get an output that looks like this:

```
     Running unittests src/main.rs (target/debug/deps/project-ac431ba9efaf64aa)

running 4 tests
test analysis::tests::test_average_score_gap_by_decade ... ok
test data_cleaning::tests::test_cleaning_sample ... ok
test graph_builder::tests::test_cosine_similarity ... ok
test graph_builder::tests::test_genre_jaccard ... ok
```

- test_cleaning_sample validates the parsed movie data to make sure it has valid values.
- test_cosine_similarity will confirm the cosine logic by returning an expected range
- test_genre_jaccard validates that it confirms correct overlap count and division
- test_average_score_gap_by_decade checks the gap calculations by decade

Moving to the results, the output folder includes: degree_centrality.csv, betweenness_centrality.csv, score_gap_by_genre.csv, score_gap_by_decade.csv, movie_clusters.txt.

The analysis revealed a lot to me. Genre-based score gaps showed that films in action, horror, and fantasy genres have the highest positive discrepancies, meaning that the audiences had rated them higher than the critics did. This can reflect how genre films can appeal to fan enthusiasm or entertainment value, which may not all the time align with critical expectations that are maybe more focused on artistic merit. Genres like documentary and drama, on the other hand, had a closer alignment between critic and user ratings.

When looking at the data over time, a stronger trend emerged that the score gap widened after the 1980s which could suggest user scores were more generous compared to critic evaluations in recent decades that could be due to the rise of users feeling more empowered to express opinions in the way films are marketed to broader audiences. The graph helped identify highly central films that are connecting diverse genres and rating patterns and the clustering algorithm grouped similar films that are reflected in coherent genre combinations and time periods. Movies form natural communities based on content and reception.

In terms of usage instructions, to build the program, I ran cargo build –release. Then running the analysis, cargo run –release – <csv_path> <genre_weight> <score_weight> <similarity_threshold>. For example, cargo run –release – imdb_top_1000.csv 0.5 0.5 0.7. Then the output files are saved to the output directory. The runtime took around a minute for me.

Overall, this project helped me apply what I've learned about graph algorithms, Rust's complex type system, and how data can be explored. It was a deep dive into the fusion of personal curiosity and analysis.

References:

Jaccard similarity log in Rust:

https://users.rust-lang.org/t/jaccard-similarity-in-rust-polars/93451

Dijkstra's algorithm explanations:

https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/

Graph theory centrality concepts:

https://en.wikipedia.org/wiki/Centrality

Petgraph crates:

https://docs.rs/petgraph/latest/petgraph/

Serialization and deserialization:

https://serde.rs/

Centrality measures:

https://towardsdatascience.com/notes-on-graph-theory-centrality-measurements-e37d2e49550a/

I used ChatGPT to refactor code because I felt that there were areas that needed concision, and I got nervous with my logic at times, so I used it as a check. I wrote the code iteratively with feedback. I also used ChatGPT to understand how to build a commit history on GitHub as I feel my knowledge on GitHub is severely lacking and wanted to ensure I had a commit history.