Project Overview

This project is supposed to take the user's anime ratings and give new anime recommendations based on them. In essence, it makes guesses based on calculations on what other anime the user would enjoy based on how similar the ratings are to other users' ratings. It uses datasets on Kaggle (anime.csv, a dataset that lists the anime on myanimelist.net and their relevant information, and rating.csv, a dataset that lists all the users and their ratings for different anime) to figure out which anime are the most aligned with the tastes of the user who have rated similarly as other people on My Anime List.

Main Parts of the Project:

Data Loading

The code starts by reading the large .csv files: anime.csv and rating.csv by utilizing Rust's csv crate. This is done in the data_processing.rs file. Each row of rating.csv is read and the data is stored by mapping each column: user_id, anime_id, and rating, into Rust fields. Looking at the dataset, we can see that -1 means that the anime was not rated by that specific user, and thus the entries with -1 were ignored/skipped. As the code loops through each rating done by each user, three key data structures were updated, including the users hashSet, the animes hashset, and the user_ratings hashmaps. The first two hashsets which keeps track of all the unique user and anime IDs that were encountered while the hashmaps store ratings per user. By the end of data loading, the program has a completed graph of which user rated which anime and by how much. A similar process was done to the anime.csv dataset which maps the anime IDs to readable anime names. Hashmaps were used for data loading as they allow for quick look-ups as given a user and anime ID as retrieving a rating has a complexity of O(1) on average.

User Input (anime name and their ratings)

After the data is completely loaded, the user is prompted to enter at least three different animes (using std::io::stdin()) they enjoyed and a score corresponding to each anime (1-10). This code is written in main.rs. The titles entered are stored in anime_title and since it is hard to enter the exact name of each anime, if the entered title doesn't exactly match any anime in the dataset, the code tries to do a fuzzy match and ask the user if their guess was correct. This uses the strsim crate's jaro_winkler function. Once the anime titles and ratings are entered into the code, they are inserted into the user_ratings map under a special user ID in order for the logic to work and treat the user as any other data point on the graph.

Similarity Calculation

The similarity calculation takes each commonly rated anime from the user and other users on My Anime List and sums the products of your ratings (sum_products) and tallies the squares (sum_rating_sq and sum_other_sq) to get magnitudes. The similarity is sum_products / (sqrt(sum_rating_sq)*sqrt(sum_other_sq)). The logic, although basic, is simple and straightforward—if two vectors are pointing in a similar direction, the output, cosine similarity, would be close to 1 and if they are super different, it would be closer to 0.

Recommendation

This is under the recommendation.rs of the project. After the similarity function/part of the project, the functions under recommendation.rs would take these similarities and generate recommendations. This is done by aggregating scores in a score map and by giving each anime a weighted score based on how similar the recommended user is to the user.

Challenges and Solutions

An issue that occurred was that a lot of niche animes would get recommended as the users might have the exact same ratings which would be inaccurate. This was fixed by adding popularity into consideration. The members of each anime are tracked so that animes with less than 5000 members would not be included in the recommendations so that the output would not be super skewed due to the low sample size. Not only that, the same few animes still showed up every single time, and thus randomization was added. Instead of just printing the top 10 recommended anime, the top 30 was taken and 10 out of the 30 was randomly selected and recommended. This introduces a lot of variety. In the beginning, I tried to implement more complicated logic to calculate similarity between the users like Pearson correlation, adjusted means, minimum overlap, but those methods implemented a lot of other issues. Another small issue that was encountered was that the anime names would not display properly due to HTML entities. HTML decoding was implemented.

Tests

Tests were implemented to test if the cosine similarity tests and calculations were correct. A few distinct scenarios were added in order to test the accuracy of the functions.

Running the Project:

1. Use cargo —release and ensure that rating.csv and anime.csv are in the depository. Using –release ensures that the program can run faster.

2. The program will prompt the user to enter their top animes with a minimum of three entries. Not only that, but each anime also has to be rated. Once entered, the output will show the recommended anime series by the project.
3. Example of output:

```
Total users loaded: 69600

Total ratings loaded: 6337241

Enter at least 3 anime titles (and up to 10) and their ratings (1-10).

Type 'done' when you are finished.

Enter anime title: naruto

Enter your rating for 'naruto': 10


Enter anime title: gintama

Enter your rating for 'gintama': 10


Enter anime title: one piece

Enter your rating for 'one piece': 10


Enter anime title: done


Recommended Anime:

Hunter x Hunter (2011)

Ookami Kodomo no Ame to Yuki

Monster

Clannad: After Story

Hajime no Ippo

Code Geass: Hangyaku no Lelouch

Suzumiya Haruhi no Shoushitsu
```

Great Teacher Onizuka

Gintama Movie: Kanketsu-hen - Yorozuya yo Eien Nare

Kizumonogatari II: Nekketsu-hen