Miles Bush and Brenden Garcia
December 8th 2019
CSCI 2270: Data Structures

movieMatch: Final Report

movieMatch is designed to take in data from polled users and connect people through their movie interests. Using google surveys, users are polled to rate 25 different movies on a 1 to 5 scale as well as provide 3 recommendations for movies that others should watch. The results of the poll are downloaded to a CSV file that is then read by the program. The user and movie data is all stored in hash tables to efficiently search and analyze the information.

DATA STRUCTURES AND METHODOLOGY

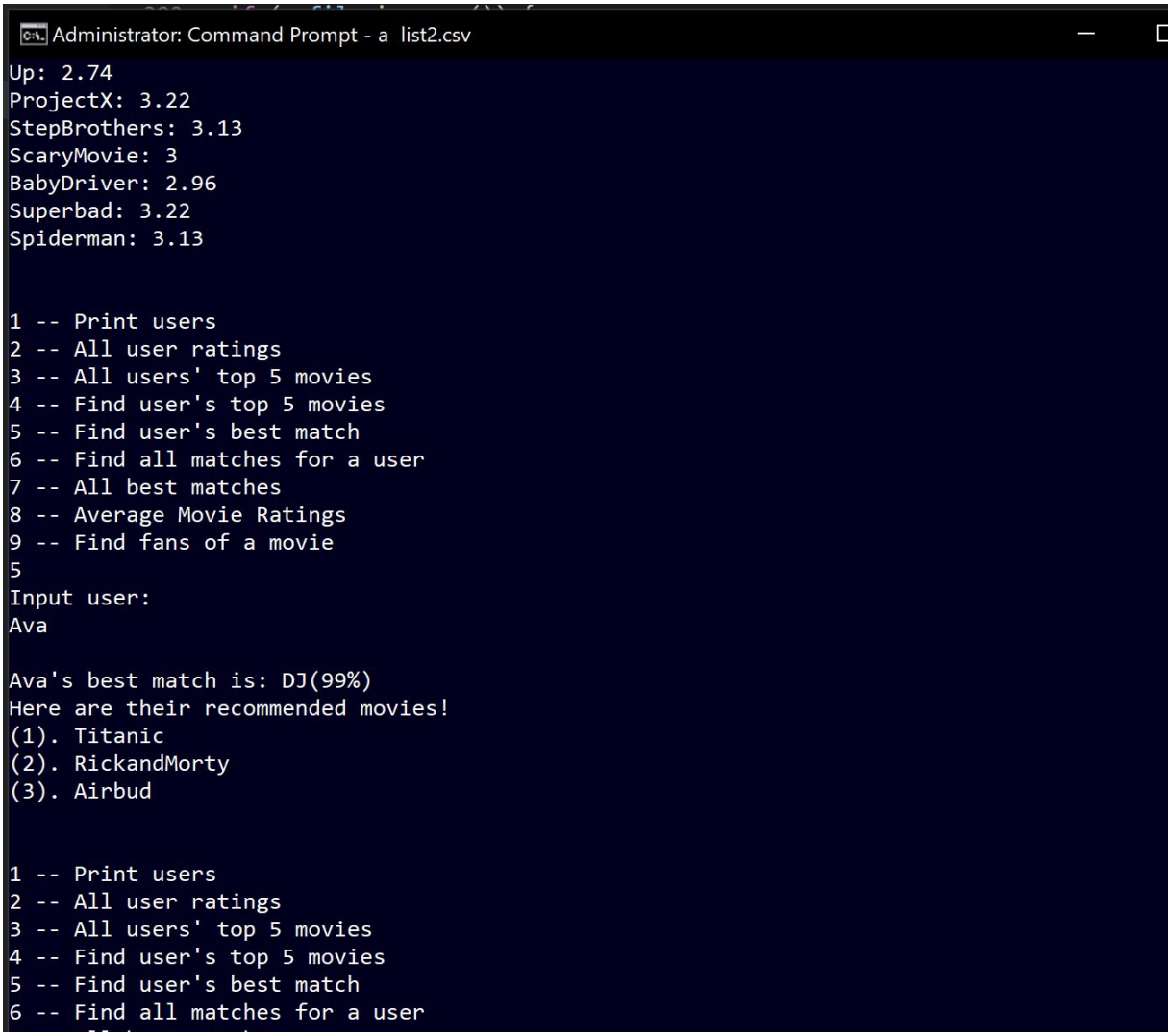
The data is stored in two separate hash tables and a few arrays. The primary hash table stores the user information and is utilized with a vector so users can be added at any time. The other hash table is used to store the information of the movies and it in the form of an array to optimize quick random access. The hash function is handled internally by the unordered_map library. Each hash table has its own unique unordered_map object that each map a string to an integer. This mapped integer is then used to access the vector/array without the need to scan through the whole array to find the object. For example, to get a users information on a specific movie, we first use the first unordered_map object to hash the name of the user and that returns an integer that corresponds to where in the vector that user is location. We next use the other unordered_map object to hash the title of the movie which returns an integer to the location of the array that that movie is stored. We then check that user's movie list at the locations specified by the hash values and we have all of that users information. The benefit of using a hash table to map to arrays is that we can quickly find users and movies without the need to iterate through the vector/array.

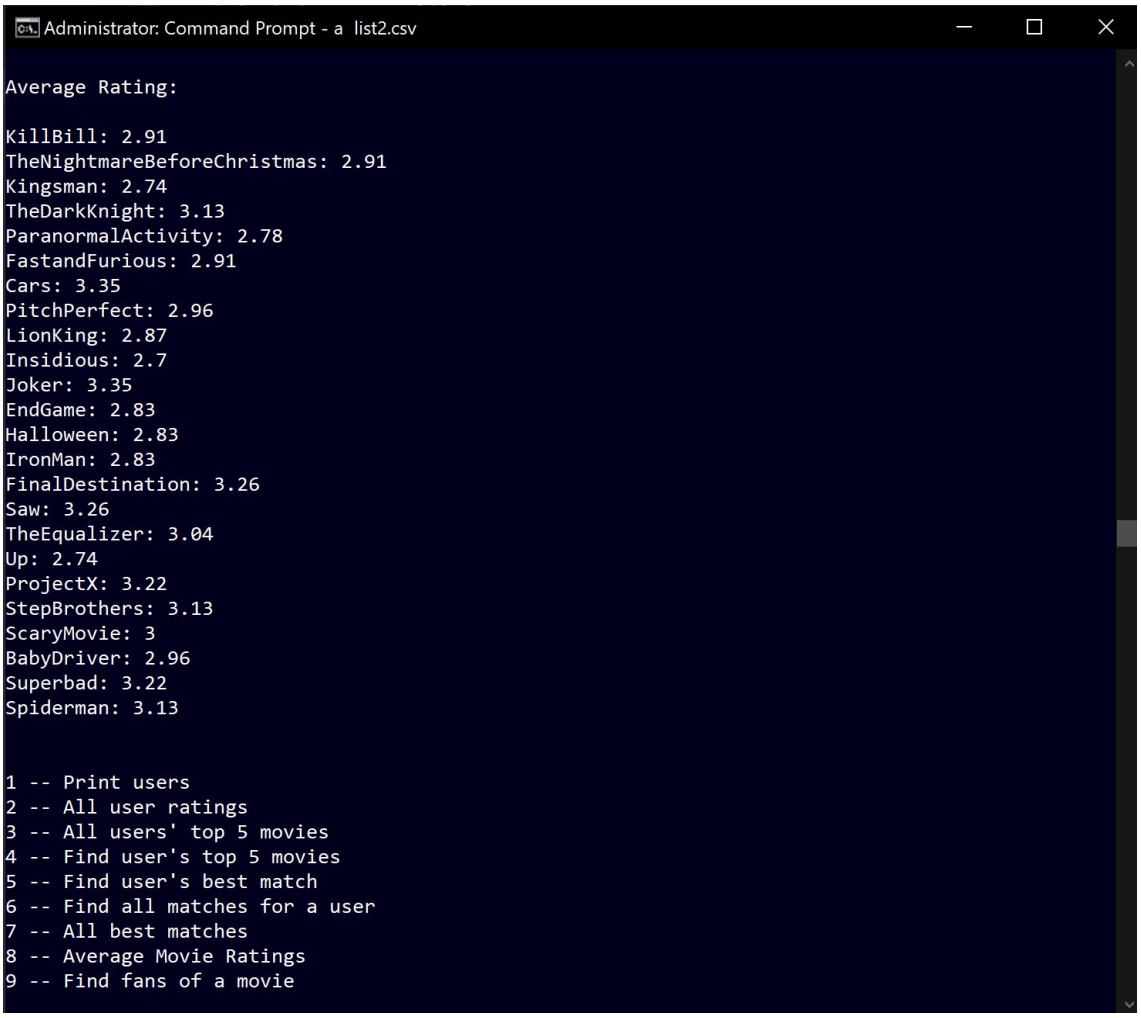
Users can be assigned a favorability rating to all other users, this determines how similar their movie interests are. To calculate this, we find each users top 5 rated movies. The hash table is then used to store the ID's (locations in the array) that those top 5 movies are stored. Then we compare user ratings to all other users in the system and generate a favorability rating. For the user's top 5 movies we take the difference in ratings, square it, and then sum the result up for all of the user's top 5 movies. That total is then converted to a percentage and shown to the user.

DATA

For the Data we decided to use google survey (which can be found here) in order to poll a large amount of people at once. The advantage with this is that we were able to store all the results of the movie polls into a single .csv file that we were able to create an f-stream to within our code. Our data consisted of three different inputs. The user's name, which is what we use to hash everything, the users ratings 1-5 for twenty five movies of mixed genres, and finally, three movies that the user would recommend to someone of similar interests.

RESULTS




Administrator: Command Prompt - a list2.csv
Average Rating:
KillBill: 2.91
TheNightmareBeforeChristmas: 2.91
Kingsman: 2.74
TheDarkKnight: 3.13
ParanormalActivity: 2.78
FastandFurious: 2.91
Cars: 3.35
PitchPerfect: 2.96
LionKing: 2.87
Insidious: 2.7
Joker: 3.35
EndGame: 2.83
Halloween: 2.83
IronMan: 2.83
FinalDestination: 3.26
Saw: 3.26
TheEqualizer: 3.04
Up: 2.74
ProjectX: 3.22
StepBrothers: 3.13
ScaryMovie: 3
BabyDriver: 2.96
Superbad: 3.22
Spiderman: 3.13
1 -- Print users
2 -- All user ratings
3 -- All users' top 5 movies
4 -- Find user's top 5 movies
5 -- Find user's best match
6 -- Find all matches for a user
7 -- All best matches
8 -- Average Movie Ratings
9 -- Find fans of a movie