CSC111 Final Project: Recommend Movies Based on Specific Attributes using Graphs

Maureen Navera

Tuesday, March 16 2021

Introduction

The COVID-19 pandemic has been raging on for more than a year now, disrupting everyone's lives and imposing on all of us a new norm, whether we like it or not. This includes practices that seem to be taken from an alternate, dystopian reality such as social distancing and wearing masks. The pandemic also meant the sudden absence of social contact. For a lot of people, this was probably the most significant among all the changes we had to endure. Having this past year spent almost entirely at home has created a dependency between people and their sources of entertainment.

Now more than ever has entertainment been so important to consumers. As the days go on, we tend to find ourselves consuming some type of media to pass time as opposed to hanging out with friends due to lockdown guidelines. Studies show that more than 25% of consumers have subscribed to at least one streaming service during the course of the pandemic (1).

But what happens when you finish a movie you enjoyed? Scrolling through various pages of streaming services to find a movie similar to the one you enjoyed can be daunting and time-consuming. This begs us to ask the following question: **how can we recommend new movies to people based on specific aspects of a movie that they enjoyed?** In other words, can we refine a search so that based on a movie, we can return a list of recommended movies that have similar aspects to that original movie, like having the same director, actors, genre, etc?

In my project, I have created an application which takes this question and answers it by asking users which elements of different movies they enjoyed, and using those answers to formulate a list of movies from IMDB's top 1000 movies dataset which apply to the user's interests.

Dataset Description

The dataset I will be using for this project is a csv file which lists IMDB's top 1000 movies of all time (2). The list contains the movie's title, a link to the movie's poster, the year it was released, its certificate, runtime, genre, IMDB rating, an overview of the movie, its media score, its director, its top 4 lead actors, the number of votes it received and its gross income. I will only be using the following columns of the dataset in this project: title, genre, IMDB rating, director and top 4 lead actors.

Computational Plan

In the file create_graph.py, I have created a Vertex and Graph class, taking the same structure as the classes we've learned this semester. I have used a Graph to represent the database of movies, with each Vertex representing a movie entry with the instance attributes title, genre (a list of genres as in the database, a movie could have more than one), The Vertex class also has the instance attributes neighbours which corresponds to the adjacent vertices, and higher_rating (which is a set that will be used to store vertices with higher ratings than this movie), director, rating (rounded to the nearest whole number), and actors (a list of the top 4 leading actors from the database). Edges are created between vertices if they fit the definition of 'similar' which i have stated in the docstring of the function 'is_similar'.

Two vertices are deemed as 'similar' if: they share at least one of the same actors, OR they share the same director, OR if they share at least one of the same genres OR if they share the same rating. If a movie is being compared with another movie of a greater rating, that second movie will be put into the first movie's instance attribute called 'higher_rating.

In the file 'menu.py', I have created an interactive GUI using Python's tkinter library. When the file 'main.py' is run, this menu is displayed and will display the instructions on how to use the Application to the user. This menu has a dropdown menu on the top left which allows users to freely choose which frame to jump to next: The Startpage, the Actors page, Director page, Rating page and Genre page, all of which are represented by a tkinter Frame. Each class is represented in the menu.py file, and each use labels to describe to the user the instructions for inputting answers for each respective category.

The actors page asks users to input a movie into an entry box. This movie is taken and cross-referenced with the graph class that has been created in the create_graph module. The list of 4 actors then show up as checkboxes for users to check off if they want to find a movie with a certain actor. A similar approach is taken with the Genre page.

The Director and the Rating page simply ask the user to input either a movie title or a number respectfully. The movie title will be cross-referenced with the graph class and a variable with that movie's director will be stored. In the Rating page, the number that has been inputted will also be stored.

These inputs from the user will be stored in another class introduced in the file 'recommend_movies.py' which includes a new class called IdealMovie. This class is much like the Vertex class we used to represent a movie from the dataset. The IdealMovie class has the instance attributes actors, director, genre, rating, and graph. In menu.py, an insance of this class is initialized in the beginning, using a graph created from the IMDB dataset and using the functions from create_graph.py to read the dataset and turn that into a graph. The other attributes are initialized to be either empty strings or lists depending on the attribute. As the user goes through the menu and fills out which categories they want, the attributes of this instance, called ideal_movie, is filled out based on what the user chooses.

The IdealMovie class has a method called _specified categories which stores which categories have been modified by the user. This helper function is used by the IdealMovie class' method find_recommendations which goes through the graph and finds one movie which fits one of the user's specification. Then the program goes through that movie's neighbour attribute in order to simplify code from going through the whole graph. The adjacent movies are then filtered whether they apply to the user's specifications, ultimately returning the final list of movie recommendations that will be printed out in the python console.

Download and Running Instructions

- Download all .py files and requirements.txt from MarkUs and move them all to one folder. After downloading the dataset form MarkUs (imdb_top_1000.zip), extract the contents of that folder and move the CSV file into the folder with the .py files and requirements.
- Now open the folder with all of the .py files in PyCharm and mark this folder as the Sources Root by right clicking Mark Directory as Sources Root
- Now install all of the libraries listed in the requirements.txt file
- Open main.py and run the file. Read the instructions that show up on the menu that is displayed.
- When you are done specifying your categories, close the application. A list of recommended movies will be displayed on the python console.

Changes from Proposal

In my original Proposal, I had planned to use a the pygame menu library, however I had decided to try something new and use tkinter to implement a GUI. I had also originally planned to initially ask the user for one movie and let them choose which aspects of that movie they were interested in. I had been suggested to implement a way to ask users the specific movie for the different categories they were interested in were . For example, for director the user would input a movie made by the director they wanted to watch, and can choose actors from another movie if they wanted too.

References

1. Balderston, Michael. "More Than 25% of Consumers Added at Least One Streaming Service During Pandemic." TVTechnology, TV Tech, 11 Aug. 2020, www.tvtechnology.com/news/more-than-25-of-consumers-added-at-least-one-streaming-service-during-pandemic.

2.https://www.kaggle.com/harshitshankhdhar/imdb-dataset-of-top-1000-movies-and-tv-shows