Movie Recommendation System

Mahek Aggarwal, Vivian Li, Daniel Veretenov

Abstract

The easy accessibility of digital streaming platforms has resulted in an abundance of movie choices for consumers, often leading to decision paralysis. This project serves to solve that problem by creating a recommendation system that can be used to suggest movies based on various factors such as genre, language, and actors. It explores the integration of MongoDB with Plotly Dash to develop an interactive movie recommendation dashboard. MongoDB is a NoSQL database that stores data in JSON-like documents. Each document consists of key-value pairs, where the keys are field names and the values can consist of various data types such as strings, numbers, or images. Plotly Dash is used with Python as the frontend to visualize and communicate our results with the user.

Introduction

We chose to create a movie recommendation system as the film industry is an important cultural part of society. In this project, we are leveraging MongoDB, a NoSQL database, to develop a movie recommendation dashboard. This interactive platform enables users to discover similar movies based on various factors like year, genre, actor, language, keyword, and other movies. MongoDB serves as the backend data repository for movie information that is obtained through The Movie Database's API (TMDB). Our goal is to design a straightforward and user-friendly dashboard for users to explore new movies and discover their upcoming favorites.

Methods

We utilized The Movie Database (TMDB) to obtain our data for this project. TMDB provides a comprehensive collection of movie metadata, including details such as titles, genres, release dates, cast, crew, and user ratings. We leveraged TMDB's API to fetch movie data programmatically.

Any movies that had missing data were skipped when loading the data. The similarity score for each movie was calculated based on TF-IDF (Term Frequency-Inverse Document Frequency) vectorization and cosine similarity and stored. The textual data that the vectorization was based on includes the genres, overview, and the keywords for each movie. TF-IDF returns a matrix where each row represents a movie, and each column represents a unique word. The values in the matrix are the TF-IDF weights of each word for each movie. The TF-IDF weight is a numerical statistic that reflects the importance of a word in a movie relative to a collection of movies. Cosine similarity is then used to compute the similarity between two vectors or movie pairs by calculating the cosine of the angle between the two vectors. For example, a value of 1 would correspond to two movies that are the same and a value of 0 would correspond to movies that are completely unalike. These similarity scores are the basis of the recommendation system.

Queries are then made to the Mongo client by establishing a connection to the MongoDB server. There are options for recommendations based on similar movies, keywords, genre, year, actor and language. The queries are returned sorted in reverse order of popularity with the most popular movie at the beginning. Each recommendation type is visualized by a different tab on the Plotly dashboard.

The frontend of this application is implemented using Plotly Dash and incorporates the six different tabs for movie-to-movie, keyword-to-movie, genre-to-movie, year-to-movie, actor-to-movie, and language-to-movie recommendations. Each tab features an input or dropdown option where the user is able to input (type) or select an option and retrieve filtered recommendation results.

Tutorial

Accessing the Dashboard: To begin, open the link to the Movie Recommendation Dashboard provided to you. Once the page loads, you will find yourself on the "Movies-to-Movies" tab by default. This tab allows you to search for similar movies based on the input you provide.

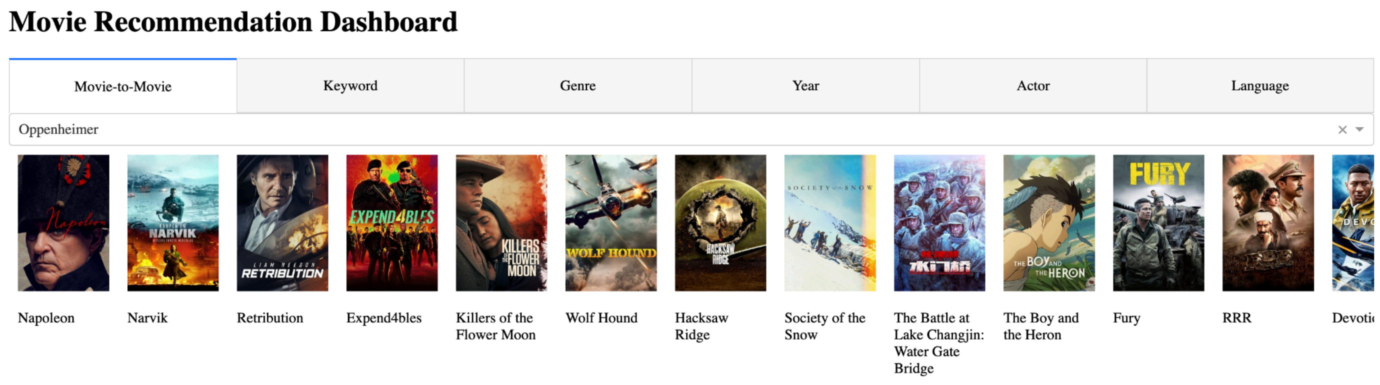
Exploring Different Tabs: At the top of the dashboard, you'll see several tabs labeled "Movies-to-Movies", "Keyword", "Genre", "Year", "Actor", and "Language". Each tab offers a different way to search for movies:

1. Movies-to-Movies: Choose the name of a movie you've liked using the dropdown and see recommended movies similar to that particular movie.
2. Keyword: This tab enables you to search for movies using specific keywords. Enter any relevant keywords related to the type of movie you're interested in, such as "romantic comedy" or "action-packed".
3. Genre: Select the genre you're in the mood for from the dropdown menu, and the dashboard will display relevant recommendations.
4. Year: Narrow down your search by selecting a particular release year. Choose the desired year from the dropdown menu to see movies released in that year.
5. Actor: Enter the name of the actor/actress, and the dashboard will suggest films they've starred in.
6. Language: This tab allows you to filter movies based on language preferences. Enter your preferred language in the box to see recommendations in that language.

Exploring Recommendations: Once you've entered your search criteria in any of the tabs, scroll horizontally to explore the recommended movies. You'll find a list of titles along with their movie posters.

Analysis

Here is an example of the movie recommendation system when the movie 'Oppenheimer' is inputted. The resulting movies from the query include Napoleon, Narvik, and Retribution.

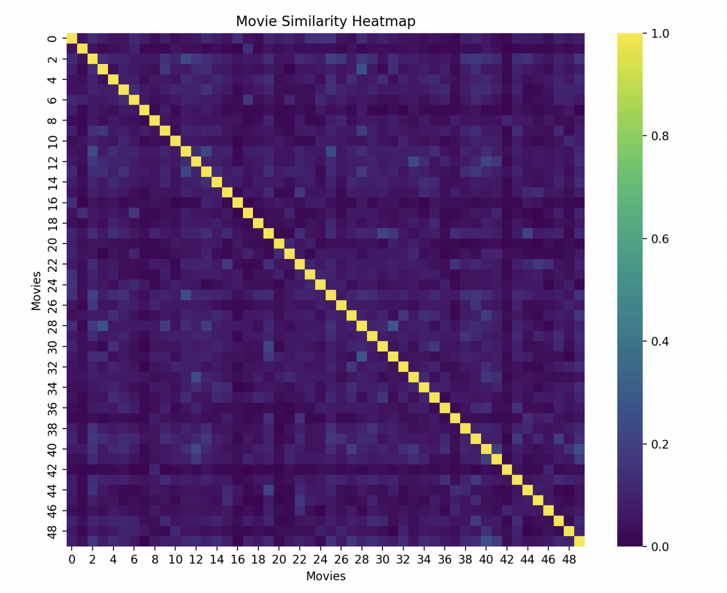

Movie Recommendation Dashboard

Movie-to-Movie	Keyword	Genre	Year	Actor	Language

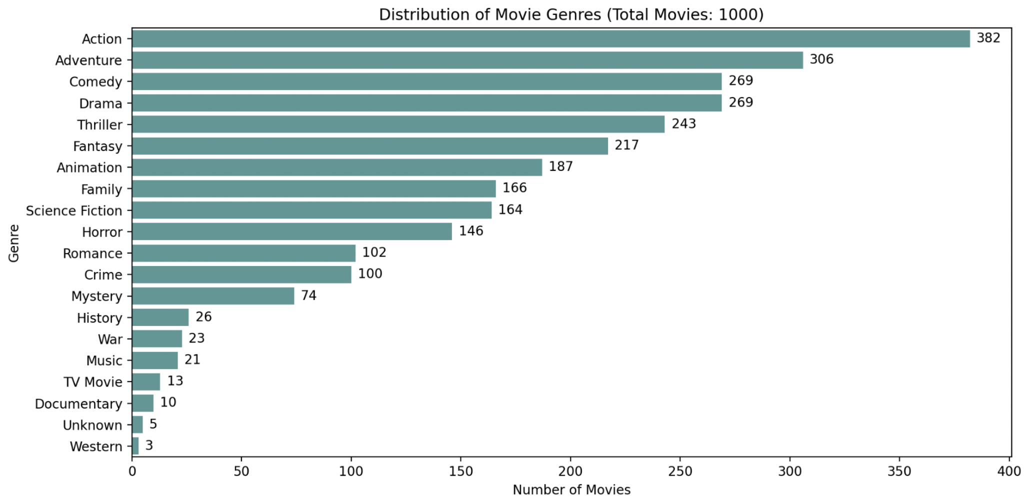
Oppenheimer

Napoleon
Narvik
Retribution
Expend4bles
Killers of the Flower Moon
Wolf Hound
Hacksaw Ridge
Society of the Snow
The Battle at Lake Changjin: Water Gate Bridge
The Boy and the Heron
Fury
RRR
Devoti

Below is an example of a heatmap of the similarity scores between 50 movies from the dataset. Both the x-axis and y-axis on the plot below represent the movie index and the color indicates the strength of the similarity between the two movies known as the similarity score. Many movies have a relatively low similarity score as seen by the purplish colored squares in the heatmap.

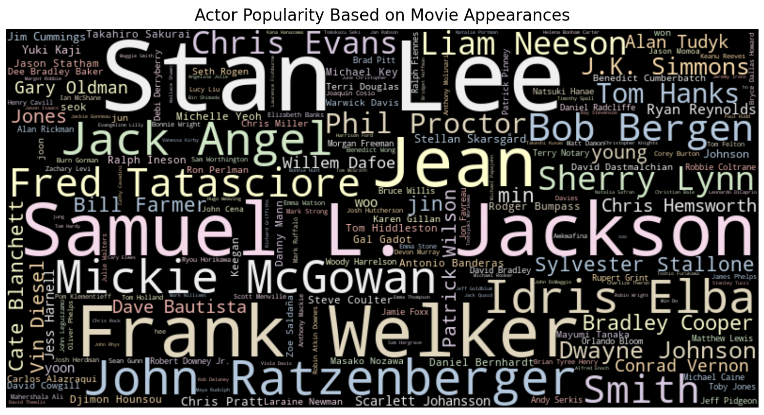


From the movie recommendation database, we can draw other analyses like the distribution of movie genres, actor popularities, and relationship between movie budget and revenue. Below are some visualizations/analyses of the exploratory data analysis done for this project.

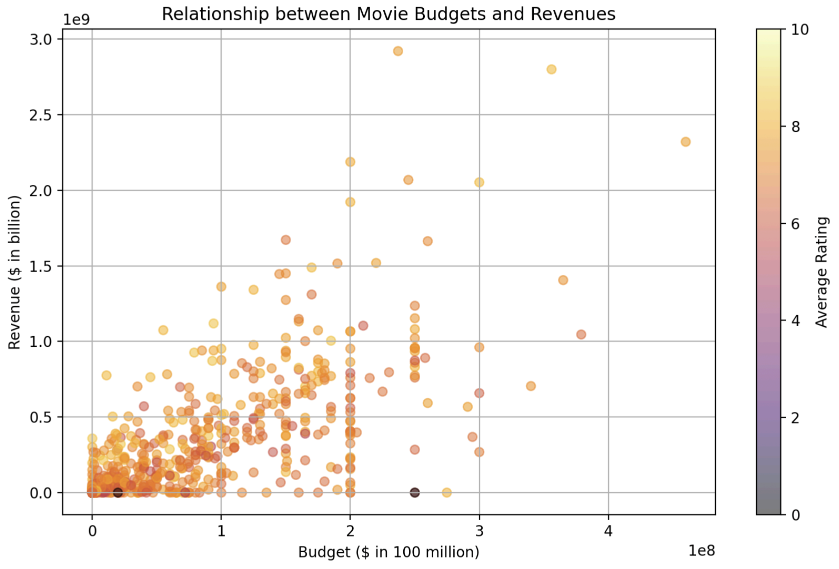


The bar chart above examines the genre distribution within a randomly selected sample of 1000 movies, providing valuable insights into cinematic diversity and genres that are most popular. The dominance of the action genre claims the largest share at 38.2%, followed closely by adventure at 30.6%. Meanwhile, genres like comedy, drama, and thriller also contribute significantly to the cinematic industry, and can offer a wide range of storytelling experiences. However, the scarcity of genres like western and documentary genres, representing only 0.3% and 1% respectively, can provide insight about genres that may not be as popular currently.

The word chart on the right highlights the popularity of actors by their frequency of appearances in movies. This method allows for a visual representation where actors with a greater number of appearances are displayed with larger font sizes, while those with fewer appearances appear smaller. The top four most prominently featured actors in the word cloud are Samuel L. Jackson, Stan Lee, Frank Welker, and Jean.



The scatter plot analysis plots the budget on the x-axis against the revenue on the y-axis and colors the points based on average rating. There seems to be a weak correlation between budget and revenue, with movies having higher budgets typically generating higher revenues. This correlation suggests that investment in a movie's production and marketing, may have the ability to significantly impact a movie's financial success or revenue. However, we observe no significant correlation between budget/revenue and the movie's average rating. Budget may be correlated to financial success, but it does not necessarily correlate with higher audience satisfaction or critical acclaim. Other factors, such as the quality of the script and direction, may play roles in determining a movie's rating. Therefore, while budget may be associated with higher commercial success, it is not associated with the overall quality/rating of the movie to viewers.



<u>Conclusions</u>

In conclusion, the application developed works well to display recommendations for movies based on user inputs. MongoDB works well as a document database to perform and return queries for the application. However, there are some limitations and possible improvements for the future that are explained further below.

Future work for this application includes combining multiple factors such as genres and actors to provide filtered recommendations based on more than one criterion. Additionally, personalizing the recommendation system to learn a user's preferences over time from movies they have enjoyed in the past or searched for is another potential improvement. Modifying the method used to calculate the movie count per tab and its functionality could enhance usability. Moreover, improving the dashboard to display more movies on one page would enhance the user experience.

Further, modifying the method used to calculate similarity scores between movies to include more textual data before performing TF-IDF vectorization could improve recommendation accuracy. Instead of using TF-IDF, another possible method involves using OpenAI embeddings to compute semantic similarities between movies. Adding the ability to click on the movie poster and be redirected to a webpage with more details about that specific movie would also enhance user interaction. Lastly, exploring the usage of MongoDB Atlas for hosting the database, which allows for automated backups and offers more scaling capabilities, could handle larger data sizes and process more data without limiting the number of movies in the database.