Movie Recommender System

CP322 – Machine Learning

Taha Amir

Professor Sukhjit Singh Sehra

190728860

April 06, 2023

Introduction

With the increasing availability of digital content, it has become more challenging for users to discover new movies that cater to their unique preferences and interests. Recommender systems have emerged as a critical tool for addressing this problem by helping users navigate the vast array of options and providing personalized recommendations based on their viewing history, preferences, or content similarity. These systems have become increasingly popular and are widely employed by streaming platforms, such as Netflix, Amazon Prime Video, and Hulu.

In this project, the aim is to develop a movie recommender system using a combination of content-based filtering, collaborative filtering, and a hybrid approach that combines both techniques. Content-based filtering focuses on the similarities between items, while collaborative filtering leverages the preferences of users with similar tastes to provide recommendations. The hybrid method combines the strengths of both approaches, offering a more comprehensive and personalized recommendation experience.

The goal is to create a system that can provide users with accurate and relevant movie recommendations, thereby enhancing their overall viewing experience and helping them discover new movies aligned with their preferences. Throughout this project, various algorithms, techniques, and data sources are explored to develop an effective recommender system, as well as evaluate its performance using appropriate metrics such as precision, recall, and F1-score.

Project Description

1. **Data Collection**:

The dataset used for this project consists of movie metadata and user ratings. The movie metadata was obtained from the 'movies_metadata.csv' file, while the user ratings were sourced from the 'ratings_small.csv' file. Both datasets are publicly available from the MovieLens dataset. The primary goal for this project was to build a movie recommender system that could effectively recommend movies to users based on their preferences and viewing history.

2. **Data Pre-processing**:

The pre-processing steps included:

- Removing unnecessary columns from the movie metadata
- Converting the 'budget' column to numeric
- Replacing empty strings in the 'original_language' column with NaN
- Converting the 'release_date' column to a datetime format
- Removing duplicate entries based on the 'id' column
- Dropping rows with missing 'title' or 'id'

For the user ratings data, the pre-processing steps included:

- Converting the 'timestamp' column to a datetime format
- Removing duplicate entries based on the 'userId' and 'movieId' columns
- Dropping rows with missing values

The cleaned movie metadata and user ratings data were saved as 'cleaned_movies_metadata.csv' and 'cleaned_ratings.csv', respectively.

3. **Exploratory Data Analysis**:

Although not included in the GitHub repository due to time constraints, exploratory data analysis would involve visualizing various aspects of the dataset, such as the distribution of movie ratings, genres, and release years. This could help identify patterns, trends, or anomalies in the data, which might influence the choice of features or modeling techniques.

4. **Feature Engineering**:

The following features were engineered for the movie recommender system:

- Average rating for each movie
- Number of ratings for each movie

These features were calculated using the user ratings data and merged with the movie metadata. Missing values for movies with no ratings were filled with zeros. The feature engineered data was saved as 'features.csv'.

The project involved building a movie recommender system using content-based filtering, collaborative filtering, and a hybrid approach. The dataset was pre-processed, and relevant features were engineered to improve the performance of the models. Improvements could be made by incorporating additional features, such as movie genres or user demographics, as well as exploring alternative modeling techniques to enhance the system's performance.

About Dataset

The dataset consists of the following files:

1. movies_metadata.csv: The main Movies Metadata file, containing information on 45,000 movies featured in the Full MovieLens dataset.
2. keywords.csv: Contains the movie plot keywords for the MovieLens movies, available in the form of a stringified JSON object.
3. credits.csv: Consists of cast and crew information for all the movies, available in the form of a stringified JSON object.
4. links.csv: Contains the TMDB and IMDB IDs of all the movies featured in the Full MovieLens dataset.
5. links_small.csv: Contains the TMDB and IMDB IDs of a small subset of 9,000 movies of the Full Dataset.
6. ratings_small.csv: The subset of 100,000 ratings from 700 users on 9,000 movies that we use for this project.

Feature Selection

To create meaningful movie recommendations for users, the first step was to select relevant features from the dataset that can be used to measure the similarity between movies or to capture user preferences. For this project, the chosen features are as follows:

1. Genres:

Genres are essential in understanding the overall theme and subject matter of a movie. Considering genre ensures that the recommendation system suggests movies that are thematically related to the ones users enjoyed in the past.

2. Original Language:

The original language of a movie can significantly influence a user's preference, as people tend to be more familiar and comfortable with movies in their native language or in languages they understand. Including the original language as a feature can help provide recommendations that are more likely to be relevant to the user's linguistic preferences.

Methodology

In this section, we will take a deeper dive into the methodology used to create the movie recommendation system. The system is built using three main techniques: content-based filtering, collaborative filtering, and a hybrid approach that combines both techniques. Here are the steps involved in each method.

1. Content-Based Filtering:

Feature extraction: The first step in content-based filtering is to extract relevant features from the movie dataset. In this case, the focus is on the movie's genre as the primary feature. Each movie is represented as a binary vector, with each element corresponding to a specific genre. If a movie belongs to a genre, the corresponding element is set to 1; otherwise, it is set to 0.

Calculating similarity: After representing the movies as binary vectors, the similarity between the input movie and all other movies is calculated using the cosine similarity metric. The cosine similarity measures the angle between two vectors, with values ranging from -1 (completely dissimilar) to 1 (identical). In this context, a higher cosine similarity value indicates a higher degree of similarity between two movies based on their genre.

Ranking and filtering: Based on the calculated cosine similarities, movies are ranked in descending order of similarity. Then the top-N movies with the highest similarity scores are filtered out and recommended.

2. Collaborative Filtering:

User-item matrix: Collaborative filtering starts by creating a user-item matrix, where each row represents a user, each column represents a movie, and the matrix entries are the user's ratings for the corresponding movies. If a user has not rated a movie, the entry is left blank or filled with a default value (e.g., zero).

Matrix factorization: To reduce the dimensionality of the user-item matrix and uncover latent features, matrix factorization techniques such as Singular Value Decomposition (SVD) or Alternating Least Squares (ALS) are applied. This process decomposes the matrix into lower-dimensional matrices representing users and movies in a latent feature space.

Predicting ratings: Using the lower-dimensional matrices, allows for predicting the user's ratings for movies they have not yet seen. These predictions are based on the dot product between the user's latent features and the movie's latent features.

Ranking and filtering: Rank the movies according to the predicted ratings and recommend the top-N movies with the highest predicted ratings for the user.

3. Hybrid Approach:

Combining scores: The hybrid approach aims to balance the strengths of both content-based and collaborative filtering techniques. To do this, the similarity scores from content-based filtering and the predicted user ratings from collaborative filtering are combined. The combined score can be calculated using various methods, such as weighted average or a more complex function that considers additional factors (e.g., popularity, recency).

Ranking and filtering: Like the previous methods, rank the movies based on their combined scores and recommend the top-N movies with the highest scores.

The methodology used in this movie recommendation system combines various techniques to provide more personalized and relevant recommendations. However, it is essential to fine-tune the model and consider additional factors to improve the system's performance further.

Challenges and Future Improvements

1. **Data Quality and Preprocessing**:

One of the challenges faced during this project was ensuring the quality of the data. Inadequate preprocessing or incomplete data might have affected the performance of the models. In the future, I could invest more time in refining the data preprocessing steps and exploring additional sources of data to improve model performance.

2. **Model Tuning**:

The models may not have been tuned optimally for the best performance. In the future, I could perform more extensive hyperparameter tuning and experiment with different model architectures to improve their performance.

3. **Evaluation Metrics**:

The performance metrics used in this project were calculated for a single user, which might not accurately represent the overall performance of the models. In future iterations, I could evaluate the models using metrics calculated across multiple users, and potentially test different metrics that better capture the performance of the recommender system.

4. **Incorporating Additional Features**:

The current models were mainly based on content-based and collaborative filtering approaches. To improve the performance of the hybrid model, I could try incorporating additional features, such as demographic information, social network data, or user-generated content like reviews and tags.

5. **Ensemble Methods and Advanced Techniques**:

I could also explore other advanced techniques, such as deep learning or ensemble methods, to potentially improve the model performance. These methods might help capture more complex relationships between users and items, leading to better recommendations.

By addressing these challenges and considering the suggested future improvements, I hope to enhance the performance of the recommender system and provide a more satisfying user experience.

Limitations

1. **Limited Processing Power**:

One of the major limitations during the development of the project was the lack of processing power. Due to constraints in computational resources, I was unable to perform extensive hyperparameter tuning, train more complex models, or process larger datasets that could have potentially improved the models' performance.

2. **Scalability**:

The limited processing power also affected the scalability of the recommender system. As the size of the dataset grows, the computational requirements for training and making recommendations increase as well. With the available resources, it was challenging to ensure that the models would scale efficiently to handle larger datasets.

3. **Time Constraints**:

Given the deadline for the project, I had limited time to explore various modeling techniques and optimizations. This restricted my ability to test alternative methods and refine the models further, which might have led to better performance.

By acknowledging these limitations, I hope to convey a realistic representation of the challenges faced during the project's development. In future iterations, I would like to explore ways to overcome these limitations, such as leveraging cloud-based resources, parallel processing techniques, or more efficient algorithms to enhance the performance and scalability of the recommender system.

Results and Experimental Analysis

In this section, we will explore the results and analysis of the movie recommender system. I tested the performance of each method using various metrics, such as precision, recall, and F1-score.

1. **Content-Based Filtering**:

The content-based filtering approach focused on providing recommendations based on the similarity between movie descriptions. I used the Nearest Neighbors model with the TF-IDF matrix to find the most similar movies to a given query. However, due to the nature of content-based filtering, I was unable to directly calculate precision, recall, and F1-score for this method, as it doesn't involve user rating predictions.

2. **Collaborative Filtering**:

The SVD model for collaborative filtering was used, which predicted user ratings based on past user behavior. The performance of this method was evaluated using precision, recall, and F1-score with a threshold of 3 and k=10. The results were as follows:

- Precision: 1.0
- Recall: 0.0011223194829104743
- F1-score: 0.0022421203663849317

These results indicate that the model was able to accurately recommend movies that users would like. However, the low recall and F1-score suggest that the model was not able to capture all relevant movies that users might enjoy.

3. **Hybrid Method**:

The hybrid method combined content-based and collaborative filtering techniques to generate recommendations. As it used the SVD model for collaborative filtering, I utilized the same predictions as in the collaborative filtering method. The performance metrics were as follows:

- Precision: 1.0
- Recall: 0.0011223194829104743
- F1-score: 0.0022421203663849317

Like the collaborative filtering method, the hybrid approach showed high precision, but low recall and F1-score. This suggests that while the hybrid method accurately recommended movies that users would like, it still struggled to identify all relevant recommendations.

The experimental analysis showed that the developed models were able to provide accurate recommendations to users. However, the low recall and F1-scores across collaborative filtering and hybrid methods indicate room for improvement. Additionally, the limitations in processing power, scalability, and time constraints affected my ability to optimize the models further. In future iterations, addressing these limitations and exploring alternative modeling techniques could lead to improved performance and a more effective movie recommender system.

Conclusion

In this project, I developed a movie recommender system using content-based filtering, collaborative filtering, and a hybrid approach that combined both techniques. The goal was to create a system capable of providing personalized and relevant movie recommendations to users. Despite the challenges faced, such as limited processing power, time constraints, and issues with initially chosen datasets, a functional recommender system was built.

The experimental analysis revealed that the models were able to accurately recommend movies that users would enjoy, as evidenced by high precision scores. However, the low recall and F1-scores for both the collaborative filtering and hybrid methods indicate that there is room for improvement in capturing all relevant recommendations. The content-based filtering approach provided recommendations based on movie similarity but could not be directly assessed using precision, recall, and F1-score metrics.

Future work could focus on addressing the limitations encountered during the project, such as refining the models, incorporating additional data sources, or exploring alternative techniques, such as deep learning-based approaches. Improving the model's scalability and processing capabilities would also enhance the overall performance and user experience.

In conclusion, the movie recommender system serves as a foundation for a more advanced and personalized recommendation engine. With continued refinement and development, I believe that the system can provide users with a valuable tool for discovering new movies tailored to their unique preferences and interests.