FlickPicks: Personalized Movie Recommendations using Collaborative Filtering

A Project Report

Submitted by:

Aarsha Anil
Marwa Abdul Razak
Navaneetha P Nambiar
Rose Benny

April 29, 2024

Vimal Jyothi Engineering College

Contents

Chapter 1

Introduction

In today's digital era, where streaming platforms offer a vast array of movies and TV shows, users often find themselves overwhelmed by choice. Amidst this abundance, discovering content that aligns with individual preferences becomes increasingly challenging. To address this issue, our project, titled "FlickPicks: Personalized Movie Recommendations using Collaborative Filtering," leverages advanced techniques in collaborative filtering to provide tailored movie recommendations to users.

The essence of our project lies in enhancing user experience by offering personalized movie suggestions based on their viewing history and preferences. Collaborative filtering, a powerful approach in recommendation systems, forms the backbone of our solution. By analyzing user interactions and identifying patterns in their movie preferences, collaborative filtering enables us to generate accurate and relevant recommendations.

Through this project, we aim to revolutionize the way users engage with movie streaming platforms. By harnessing the collective wisdom of user behaviors, FlickPicks empowers individuals to discover new and exciting content that resonates with their tastes and interests. In the following sections, we delve deeper into the methodology, implementation, and results of our collaborative filtering-based movie recommendation system.

Chapter 2

Dataset

The MovieLens Small dataset is a subset of the larger MovieLens dataset, which is a popular movie recommendation dataset used for research and evaluation purposes in the field of recommendation systems. The Small dataset is designed for educational and experimentation purposes, containing a smaller number of users, movies, and ratings compared to the full dataset. It provides a more manageable size for testing algorithms and prototyping recommendation models.

Key features of the MovieLens Small dataset include:

- **Users:** The dataset contains information about a set of users who have rated movies. Each user is represented by a unique identifier.
- **Movies:** It includes a collection of movies available on the platform. Each movie is identified by a unique ID and may have associated metadata such as title, genre, release year, etc.
- **Ratings:** Users in the dataset have rated movies on a predefined scale, typically ranging from 1 to 5 stars. Ratings are provided for the interactions between users and movies, forming the basis for recommendation algorithms to learn user preferences.
- **Sparse Interactions:** Like many real-world recommendation datasets, the interactions between users and movies are often sparse, meaning that not every user has rated every movie. This sparsity presents challenges for recommendation algorithms to make accurate predictions and generate relevant recommendations.
- **Data Format:** The MovieLens Small dataset is commonly available in CSV (Comma-Separated Values) format, making it easy to load and analyze using various programming languages and data analysis tools.

Overall, the MovieLens Small dataset serves as a valuable resource for researchers, students, and practitioners interested in exploring recommendation algorithms, collaborative filtering techniques, and personalized movie recommendations. It provides a manageable yet representative sample of user-item interactions, facilitating experimentation and evaluation of recommendation systems.

This dataset (ml-latest-small) describes 5-star rating and free-text tagging activity from MovieLens, a movie recommendation service. It contains 100836 ratings and 3683 tag applications across 9742 movies. These data were created by 610 users between March 29, 1996 and September 24, 2018. This dataset was generated on September 26, 2018.

Users were selected at random for inclusion. All selected users had rated at least 20 movies. No demographic information is included. Each user is represented by an id, and no other information is provided.

The data are contained in the files links.csv, movies.csv, ratings.csv and tags.csv.

Chapter 3

Collaborative Filtering

Collaborative filtering is a popular and effective technique used in recommendation systems to provide personalized recommendations to users. It leverages the collective wisdom of a community of users to make predictions or recommendations.

The fundamental idea behind collaborative filtering is to identify patterns or similarities among users' preferences and behaviors. Instead of relying on explicit knowledge about items (e.g., movies) or users, collaborative filtering algorithms analyze past interactions between users and items to infer preferences and make predictions.

There are two main approaches to collaborative filtering:

1. **User-Based Collaborative Filtering:**
   - In user-based collaborative filtering, recommendations are made to a target user based on the preferences of similar users.
   - The algorithm identifies users with similar rating patterns or tastes to the target user and recommends items that these similar users have liked or rated highly.
   - User-based collaborative filtering is intuitive and easy to implement. However, it can suffer from scalability issues as the number of users grows, and it may not perform well in the presence of sparse data.
2. **Item-Based Collaborative Filtering:**
   - In item-based collaborative filtering, recommendations are made by identifying items similar to those the target user has already interacted with or rated positively.

- The algorithm computes similarities between items based on the ratings they receive from users. Items with high similarity to those already liked by the user are recommended.
- Item-based collaborative filtering tends to perform well with sparse data and is computationally efficient. It also avoids the scalability issues associated with user-based approaches.

Collaborative filtering methods do not require explicit knowledge about items or users' preferences, making them suitable for various recommendation scenarios. They excel in capturing user preferences, even in the absence of detailed item descriptions or user profiles. However, challenges such as the cold-start problem (difficulty in making recommendations for new users or items) and scalability issues with large datasets require careful consideration and algorithm design.

Benefits of Collaborative Filtering

1. **Personalized Recommendations:** Collaborative filtering provides personalized recommendations to users based on their past interactions and preferences. By identifying similarities among users or items, it can suggest items that align with an individual's tastes and preferences.
2. **No Need for Item Details:** Unlike content-based filtering, which relies on detailed item attributes or features, collaborative filtering only requires user-item interaction data, such as ratings or purchase history. This makes it suitable for scenarios where item descriptions are limited or unavailable.
3. **Serendipitous Discovery:** Collaborative filtering can lead to serendipitous discoveries by recommending items that users may not have considered or discovered on their own. By identifying patterns in user behavior, it can surface hidden gems or niche items that align with a user's preferences.

4. **Scalability:** Collaborative filtering algorithms can scale effectively to large datasets and user bases. They leverage collective wisdom from a community of users to make recommendations, making them suitable for platforms with a vast number of users and items.
5. **Adaptability:** Collaborative filtering can adapt to changes in user preferences or trends over time. As users interact with the system and provide feedback, the recommendations can evolve to reflect shifting tastes and preferences.
6. **Cold Start Handling:** Collaborative filtering can handle the cold-start problem, where new users or items have limited interaction data. By leveraging similarities among users or items, it can make reasonable recommendations even for users with sparse interaction histories or newly added items.
7. **Diversity:** Collaborative filtering algorithms can promote diversity in recommendations by considering a wide range of user preferences. By identifying similarities among users with diverse tastes, it can recommend items that appeal to different segments of the user base.

Chapter 4

Methods and Techniques

We developed a movie recommendation system using collaborative filtering techniques. The process involved preprocessing the MovieLens dataset, implementing a collaborative filtering model using TensorFlow and Keras, training the model, and evaluating its performance.

4.1 Data Loading and Preprocessing

1. First load movie ratings data from a CSV file named `ratings.csv` into a Pandas DataFrame called `rating`.
2. Unique user and movie IDs are extracted and encoded to index values for modeling purposes.
3. Ratings are normalized between 0 and 1 to facilitate training.

4.2 Model Architecture

1. The collaborative filtering model architecture is defined using Keras.
2. It comprises embedding layers for users and movies to capture latent features.
3. User and movie embeddings are flattened and combined using a dot product operation.
4. Multiple dense layers with ReLU activation functions learn non-linear mappings between embeddings.
5. Mean squared error loss function and the Adam optimizer are used for optimization.

4.3 Model Training

1. The model is trained using a random train-test split of the dataset.
2. Training is performed for 10 epochs with a batch size of 64.

4.4 Model Evaluation and Saving

1. After training, the model's performance can be evaluated using various metrics.
2. The trained model is saved in the h5 format for future use.

4.5 Movie Recommendation Generation

1. Given a user ID, personalized movie recommendations are generated.
2. Movies not watched by the user are identified, and ratings for these movies are predicted using the trained model.
3. The top-rated movies are recommended to the user based on the predicted ratings.

Chapter 5

Model Training

5.1 Deep Learning Components in Collaborative Filtering Model

5.1.1 Embedding Layers

The model includes embedding layers for both users and movies. Embeddings are dense vector representations of categorical variables (user IDs and movie IDs in this case) learned during training. These embeddings capture latent features of users and movies, enabling the model to capture complex relationships and preferences.

5.1.2 Dense Layers

The embeddings of users and movies are flattened and then combined using a dot product operation to calculate the predicted ratings. The dot product is followed by a series of dense (fully connected) layers with ReLU activation functions. These dense layers allow the model to learn non-linear mappings between user and movie embeddings, potentially capturing more complex interactions.

5.1.3 Loss Function

The model is trained using mean squared error (MSE) loss function, which measures the difference between the predicted ratings and the

actual ratings provided in the dataset. Minimizing this loss function during training aims to improve the model's ability to predict ratings accurately.

5.1.4 Optimization

The model is optimized using the Adam optimizer, a popular optimization algorithm for training neural networks. Adam adapts the learning rates for each parameter during training, allowing for faster convergence and improved performance.

Overall, this collaborative filtering model utilizes deep learning techniques to learn representations of users and movies from raw data and make personalized movie recommendations based on learned embeddings and interactions.

Chapter 6

Interface

We developed a Streamlit app for movie recommendations. The app allows users to input their ID and receive personalized movie recommendations based on their preferences and interactions. It utilizes a pre-trained deep learning model for collaborative filtering and provides an interactive interface for users to explore recommendations.

1. Data Loading

- **Ratings Data**: A dataset containing user ratings for various movies. Each entry typically includes the user ID, movie ID, and the rating given by the user.
- **Movie Data**: Additional information about the movies, such as title, genres, etc., used for displaying movie details in the recommendation results.

2. Data Preprocessing

- **Mapping IDs to Indices**: Categorical variables like user IDs and movie IDs are mapped to integer indices using dictionaries (`user2user_encoded`, `movie2movie_encoded`).
- **Adding Encoded Indices to Ratings Data**: Additional columns containing the encoded indices of users and movies are added to the ratings data.

3. Model Loading

- A pre-trained deep learning model is loaded into memory. This model is trained for collaborative filtering tasks, predicting user ratings for movies based on interactions.

4. Recommendation Generation

- **User Input**: The user provides their ID through a user interface to identify the user for whom recommendations are to be generated.
- **Finding Movies Not Watched**: Movies not yet watched by the user are identified by comparing the dataset with movies the user has rated.
- **Encoding Movies and Users**: The remaining movies are encoded using the created mappings, and the user ID is similarly encoded.
- **Predictions**: The model predicts ratings for user-movie pairs by passing the encoded indices through the neural network model.
- **Top Recommendations**: Movies with the highest predicted ratings are selected as recommendations.

5. Displaying Recommendations

- Recommended movies are displayed to the user through a user interface. A Streamlit web application provides an interactive interface for users to input their ID and view recommendations.

6. User Interaction

- Users interact with the system by providing their ID and requesting recommendations. The system responds by displaying top-rated movie recommendations tailored to the user's preferences.

Overall, the process involves loading and preprocessing data, utilizing a pre-trained deep learning model, and presenting personalized movie recommendations to the user through an interactive interface.

Chapter 7

Result

The output of the Streamlit app showcases personalized movie recommendations for users based on their input ID.

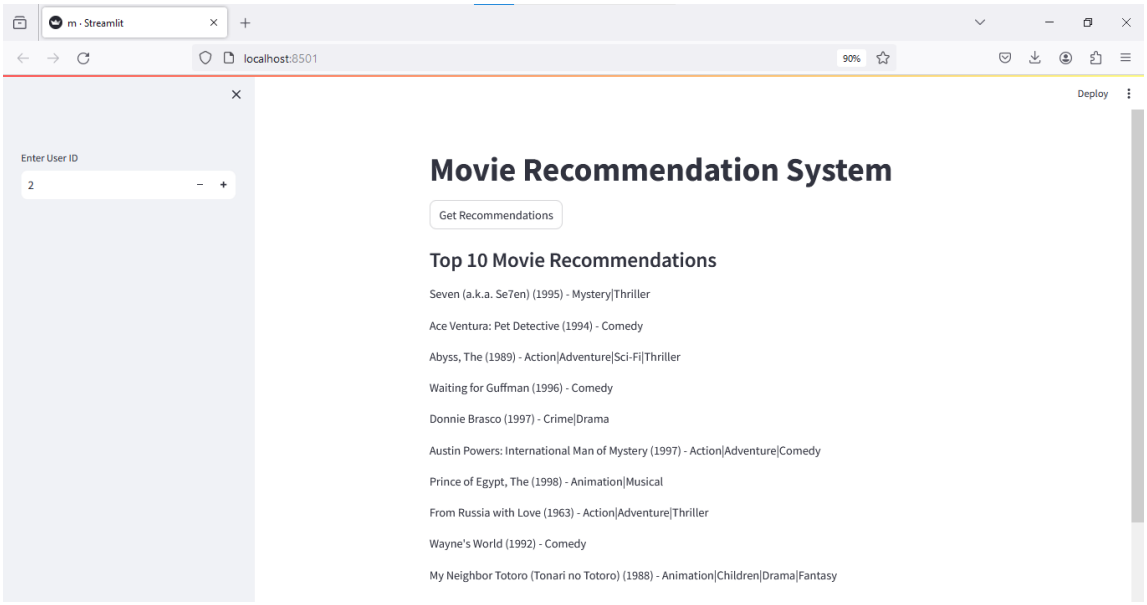

Figure 7.1: Result

Chapter 8

Conclusion

Collaborative filtering techniques, particularly those implemented using deep learning models, offer powerful solutions for personalized recommendation systems. By leveraging user interactions and preferences, these systems can provide tailored recommendations, enhancing user engagement and satisfaction. Through the utilization of embeddings and neural networks, collaborative filtering models can capture complex patterns and relationships in user-item interactions, enabling accurate predictions and effective recommendation generation. As demonstrated in this project, collaborative filtering techniques, when coupled with appropriate data preprocessing and model training, can yield valuable insights and improve user experiences in various domains, including e-commerce, entertainment, and content streaming platforms. Moving forward, continued advancements in deep learning methodologies and data collection strategies are expected to further enhance the capabilities and performance of collaborative filtering recommendation systems, paving the way for more personalized and impactful user experiences in the digital landscape.