Learning Machine

Third Assignment

Writer:

Mohammad Rezaei Kalantary

Student number:

401422087

Professor:

Dr Hadi Farahani

Shahid Beheshti University
Faculty of Mathematics
Spring 2023

Contents

Abstract

This project deploys a machine learning model on Hugging Face Spaces for a movie recommendation task. The recommender system utilizes cluster-based and recommendation-specific methods to generate personalized movie recommendations. The user-friendly interface, designed using Gradio, allows users to input movie titles and select the recommendation method. Deploying on Hugging Face Spaces enables easy access, collaboration, and reproducibility, enhancing the dissemination and utilization of the movie recommender system.

Keywords: Movie recommendation systems, machine learning, Hugging Face Spaces, cluster-based methods, content-based Filtering, recommendation-specific methods, user interface, personalized recommendations, collaboration, reproducibility.

Introduction

Movie recommendation systems have become an integral part of the entertainment industry, aiming to provide personalized movie suggestions to users based on their preferences. With the vast amount of movies available today, users often rely on recommendation systems to discover new content tailored to their tastes. These systems leverage machine learning algorithms to analyze user data and movie characteristics to generate relevant and engaging recommendations.

In this project, our objective is to deploy a machine learning model on Hugging Face Spaces for a movie recommendation task. We will utilize the Movies Dataset, a comprehensive collection of data on 45,000 movies, which includes information such as budget, revenue, release dates, languages, production countries, and companies for each movie. This dataset, obtained from The Movie Database (TMDb) API, provides us with a rich source of movie-related information that we can leverage to build an effective recommender system.

The primary goal of our project is to develop a movie recommender system that can accurately recommend movies based on user preferences. By analyzing the dataset and employing machine learning techniques, we aim to create a model that can understand user tastes and preferences, and provide personalized recommendations that align with their interests.

To achieve this objective, our project will primarily focus on three approaches: content-based filtering, Collaborative filtering and cluster-based methods. These methods offer effective strategies to generate accurate and diverse movie recommendations based on user preferences.

Content-based filtering involves recommending movies to users based on the similarity of movie attributes. By analyzing the characteristics of movies, such as genre, actors, directors, and keywords, we can identify patterns and similarities between movies. This approach allows us to recommend movies that share similar features with the ones a user has enjoyed in the past. Content-based filtering is particularly useful in capturing user preferences and providing personalized recommendations.

Collaborative filtering, on the other hand, involves recommending movies to users based on their similarity to other users' preferences. By analyzing the past movie ratings and preferences of users, we can identify users with similar tastes and recommend movies that have been highly rated by those similar users. Collaborative filtering leverages the wisdom of the crowd and can provide recommendations even for users with limited historical data.

Cluster-based methods, on the other hand, involve grouping movies or users into clusters based on their similarities. By leveraging clustering techniques, we can identify groups of movies that exhibit similar characteristics or appeal to similar audiences. This approach allows us to make recommendations based on the preferences and characteristics of other movies or users within the same cluster. Additionally, we can recommend movies from different clusters to introduce new content to the user or identify movies located at the center of the clusters that may have broader appeal.

To enhance user experience and engagement, we will design a user-friendly interface using Gradio, a popular Python library for creating interactive interfaces in machine learning projects. This interface will allow users to interact with the recommender system, input their movie preferences, and receive personalized recommendations based on their choices.

Finally, we will deploy our trained machine learning model and user interface on Hugging Face Spaces. Hugging Face Spaces provides a platform for showcasing and sharing machine learning models, enabling others to easily interact with and benefit from our movie recommender system. By deploying our system on Hugging Face Spaces, we aim to make our model readily accessible to users and contribute to the wider machine learning community.

In conclusion, our project focuses on deploying a machine learning model on Hugging Face Spaces for a movie recommendation task. By leveraging the Movies Dataset and employing various recommendation methods, we aim to build an effective movie recommender system that provides personalized and engaging movie recommendations to users.

1 Methodology

To implement the movie recommender system, we employed a combination of content-based filtering, collaborative filtering, and cluster-based methods. Each method has its unique approach to generating movie recommendations based on user preferences and movie characteristics. In this section, we will provide detailed explanations and insights into each method, along with the rationale behind the design choices and strategies employed.

1. **Content-Based Filtering**

    - Preprocessing: We started by cleaning the movie titles to remove special characters and convert them to lowercase. This step ensures consistency and improves the accuracy of similarity calculations. Additionally, we cleaned the 'genres' column, converting it from a string representation to a list of genres. By transforming the data into a more structured format, we can extract meaningful information from the genres for recommendation purposes.
    - TF-IDF Vectorization: The TF-IDF (Term Frequency-Inverse Document Frequency) vectorization technique was employed to represent the movie titles as a numerical feature vector. This approach calculates the importance of each word in the movie titles based on its frequency in a specific title and its rarity across the entire dataset. TF-IDF helps capture the uniqueness of each movie title and allows us to calculate the similarity between movies based on their textual representations.
    - Movie Search: Given a movie title, we implemented a search function that utilizes cosine similarity to identify the most similar movies in the dataset. By comparing the TF-IDF vector of the input movie title with the TF-IDF vectors of all movies, we can compute the cosine similarity scores. Higher cosine similarity scores indicate a closer match between movies, enabling us to recommend movies that are similar in terms of genre, actors, directors, and keywords.

Rationale: The content-based filtering method was chosen to provide personalized recommendations based on the attributes of movies that users have already enjoyed. By leveraging the textual information from movie titles and genres, we can capture user preferences and recommend movies that share similar characteristics. This approach allows for fine-grained recommendations based on specific movie attributes, enhancing the relevance and personalization of the recommendations.

2. **Collaborative Filtering**

   - Data Preprocessing: We performed data preprocessing by cleaning and merging the 'ratings' and 'links' datasets. This step involved removing any unnecessary columns and ensuring consistency in the movie IDs and user ratings. Merging these datasets allows us to access the movie ratings and corresponding movie IDs for collaborative filtering.
   - Finding Similar Movies: To identify similar movies, we focused on finding similar users who have rated the input movie positively. We selected users who rated the movie above a certain threshold to ensure that the recommendations are based on positive feedback. By identifying similar users, we can extract the movies they rated highly, indicating a potential preference overlap with the input movie.
   - Recommendation Generation: Once we have obtained the movies rated highly by similar users, we merged this information with the Movies dataset to access additional movie details such as release dates, titles, and genres. Sorting the recommended movies based on the release year allows us to provide more recent and up-to-date recommendations to the users.

   Rationale: Collaborative filtering is a widely used method for recommendation systems as it leverages the collective wisdom of users with similar preferences. By analyzing the ratings given by similar users, we can infer a user's taste and recommend movies that have been highly rated by those similar users. This approach adds a serendipity factor to the recommendations, introducing users to movies they may not have discovered otherwise.

3. **Cluster-Based**

   - Preprocessing: In the cluster-based approach, we focused on grouping movies based on their genre matrix. We created a genre matrix by one-hot encoding the 'genres' column in the Movies dataset. This matrix represents the presence or absence of each genre for each movie. By encoding the genres, we capture the characteristicof each movie, allowing us to group similar movies together.
   - Cluster Generation: To create clusters, we utilized the KMeans algorithm. The number of clusters was set to 50 based on experimentation and finding a balance between too few and too many clusters. KMeans iteratively assigns movies to clusters based on their genre matrix, aiming to minimize the within-cluster sum of squares. This process helps identify groups of movies that share similar genre profiles.
   - Recommendation Generation: Given three input movie titles, we determined their corresponding cluster labels. By identifying movies belonging to the same clusters as the input movies, we can recommend movies that share similar genre profiles. The recommended movies are then sorted based on release dates, genres,

and other relevant factors to provide a diverse and engaging set of recommendations.

Rationale: The cluster-based approach complements the content-based and collaborative filtering methods by offering a different perspective on similarity. By grouping movies into clusters based on their genre profiles, we can recommend movies that share similar characteristics, even if they may not be directly related to the input movies. This approach helps introduce users to new and diverse content, expanding their movie-watching experience.

In our methodology, we combined content-based filtering, collaborative filtering, and cluster-based methods to provide a comprehensive and effective movie recommender system. The content-based filtering method focuses on recommending movies based on similarities in movie titles and genres, capturing individual preferences. Collaborative filtering leverages the ratings and preferences of similar users to offer diverse recommendations. The cluster-based method groups movies based on genre profiles, enabling the discovery of new and relevant movies.

The rationale behind employing multiple methods is to leverage their strengths and enhance the recommendation process. Content-based filtering captures specific user preferences and recommends movies with similar attributes. Collaborative filtering introduces serendipity and diversity by considering the preferences of similar users. Cluster-based recommendations offer an alternative perspective by suggesting movies with similar genre profiles, expanding the range of recommendations.

By combining these approaches, we aim to provide users with personalized and relevant movie recommendations that cater to their unique preferences while also introducing new and diverse content. The chosen design choices and strategies were motivated by the goal of creating an engaging and effective movie recommender system.

2 Dataset Preprocessing

In the preprocessing stage, we performed several steps to clean and prepare the Movies dataset for further analysis and recommendation generation. Here is an explanation of the preprocessing steps applied:

1. **Data Cleaning**

    - Cleaning Movie Titles: We defined a cleaning function, clean title(), to remove any special characters from the movie titles using regular expressions. This step ensures consistency and improves the quality of the textual data.
    - Dropping Irrelevant Rows: We dropped specific rows from the dataset using the drop() function. The rows with indices 19730, 29503, and 35587 were removed, as they were found to be irrelevant or containing incomplete data.
    - Cleaning Columns: Several columns such as 'genres', 'production companies', 'production countries', and 'spoken languages' contained string representations of lists. We applied the literal eval() function to convert these string representations to actual lists. Additionally, we used a lambda function to extract the names of the items within these lists, resulting in cleaner and more structured data.

2. **Renaming and Numeric Conversion**

- Renaming 'id' Column: We renamed the 'id' column to 'movieId' using the rename() function. This step improves clarity and aligns the column name with its purpose in the dataset.
- Numeric Conversion: The 'movieId' column was converted to numeric values using pd.to numeric() to ensure proper data type and compatibility for future calculations and merging with other datasets.

3. **TF-IDF Vectorization**

   - Creating the TF-IDF Vectorizer: We utilized the TfidfVectorizer from the scikit-learn library to generate TF-IDF vectors from the cleaned movie titles. The vectorizer considers unigrams and bigrams (ngram range=(1,2)) to capture the importance of both individual words and word combinations in the movie titles. This vectorization step helps quantify the relevance and significance of each word in the titles for subsequent similarity calculations.

4. **Handling Ratings Data**

   - Loading and Sorting Ratings: We loaded the 'ratings' dataset using pd.read csv() and dropped the 'timestamp' column since it was not necessary for our recommender system. We sorted the ratings based on the second column (typically 'userId') to ensure consistency in the ratings data.
   - Merging with Links Data: We loaded the 'links' dataset and dropped the 'imdbId' column. Then, we merged the sorted ratings dataset with the links dataset on the 'movieId' column to obtain the movie ratings and corresponding movie IDs. This step helps establish the connection between movie ratings and the Movies dataset for collaborative filtering.

Rationale: The preprocessing steps performed on the Movies dataset were crucial for cleaning and structuring the data to facilitate subsequent analysis and recommendation generation. By cleaning movie titles, handling missing values, and converting columns to appropriate data types, we ensure the integrity and quality of the dataset. Additionally, the TF-IDF vectorization allows us to represent movie titles as numerical features, enabling similarity calculations for content-based filtering. The handling of ratings data and merging it with movie information is essential for collaborative filtering, where we leverage user ratings to generate recommendations.

Overall, these preprocessing steps set the foundation for building an effective movie recommender system, enabling us to extract valuable insights and provide personalized movie recommendations to users based on their preferences.

3 Recommender System Implementation

The implementation of our movie recommender system follows a workflow that involves collecting user preferences, generating recommendations based on the selected method (collaborative, cluster, or content-based), and evaluating the system's performance using appropriate metrics. Additionally, we designed a user-friendly interface using the Gradio library to enhance the user experience and interaction.

1. **User Preferences**

- Collecting User Input: We collect user preferences by prompting them to enter the titles of three movies. These movie titles serve as the basis for generating personalized recommendations. The user can provide the movie titles through the interface's input fields.

2. **Recommender Procedure**

- Method Selection: The user selects one of the three available recommendation methods: collaborative, cluster, or content-based.
- Prediction Function: We have a prediction function, predict(), that takes the three movie titles and the selected method as inputs. Based on the chosen method, the function executes the corresponding recommendation logic.
- Collaborative Filtering: If the collaborative method is selected, the function performs a search for each movie title to retrieve the most similar movies using the search() function. Then, it finds similar movies based on user ratings using the find similar movies() function. The recommended movies from each title are concatenated into a single dataframe.
- Cluster-Based Filtering: For the cluster method, the function performs a search for each movie title using the search cluster() function. It then calls the cluster based recommender() function, passing the titles of the search results as inputs, to generate recommendations based on clustering.
- Content-Based Filtering: If the content-based method is chosen, the function performs a search for each movie title using the search content() function. It calls the content based recommendation() function, passing the titles of the search results as inputs, to generate content-based recommendations.
- Invalid Method Handling: If an invalid recommendation method is selected, the function returns an error message indicating the available options.

3. **Similarity Measures**

- Collaborative Filtering: The similarity between movies in collaborative filtering is determined by analyzing the ratings patterns of similar users. The function finds similar users who have rated the input movies positively, and then calculates the similarity based on the movies highly rated by those users.
- Cluster-Based Filtering: In the cluster-based method, the similarity between movies is based on their genre profiles. The function assigns movies to clusters using KMeans clustering based on the genre matrix. It recommends movies that belong to the same cluster as the input movies, ensuring similarity in genre characteristics.
- Content-Based Filtering: Similarity in content-based filtering is determined by analyzing the TF-IDF vectors of movie titles. The function calculates the cosine similarity between the input movie titles and the TF-IDF vectors of the entire dataset. Movies with higher cosine similarity scores are considered more similar and recommended.

4 UI UX Design

For designing the user interface, we utilized the Gradio library. Gradio offers a simple and intuitive way to create interactive interfaces for machine learning models. The library seamlessly integrates with popular Python frameworks and provides customization options to enhance the user experience.

1. **Choice of Gradio**

   - Simplicity and Versatility: Gradio provides a straightforward API that allows us to define the prediction function, input/output types, and other interface parameters easily. It supports various input types, such as text, images, and audio,and provides flexibility in defining the output format.
   - Integration with Machine Learning Models: Gradio seamlessly integrates with machine learning models, enabling us to connect our recommendation algorithms with the user interface. It simplifies the process of deploying the recommender system.

2. **Design Considerations**

   - Input Fields: The interface includes three input fields where users can enter the titles of their preferred movies. This input serves as the basis for generating personalized recommendations.
   - Method Selection: The interface offers a radio button group that allows users to select one of the three available recommendation methods: collaborative, cluster, or content-based. This selection determines the algorithm used to generate recommendations.
   - Output Format: The output of the interface is a dataframe containing the recommended movies' titles, release years, and genres. This format provides users with essential information about the recommended movies.
   - Example Inputs: To assist users and provide a better understanding of the input format, the interface includes example inputs. These examples demonstrate how to enter movie titles and select the recommendation method.
   - Theming and Description: The interface incorporates theming to enhance visual appeal and provides a description that explains the functionality and purpose of the recommender system. The description sets user expectations and clarifies the limitations of the system.

3. **Interactivity and Customization**

   - Real-time Recommendations: Gradio allows for real-time recommendations, providing instant results based on user input. Users can see the recommendations change dynamically as they enter different movie titles.
   - Flagging Options: To gather user feedback, the interface includes flagging options, such as ”Good Prediction” and ”Bad Prediction.” Users can provide feedback on the accuracy or relevance of the recommendations, allowing for continuous improvement of the system.

- Customization: Gradio offers customization options for the interface's appearance, such as theming and layout. These features enable us to create a visually appealing and user-friendly interface that aligns with the recommender system's purpose and branding.

By leveraging the capabilities of Gradio, we have designed a user-friendly interface that allows users to interact with the recommender system effortlessly. The interface provides an intuitive way to input movie preferences, select the recommendation method, and view personalized recommendations, enhancing the overall user experience and engagement.

5 Model Deployment on HuggingFace Spaces

Deploying the final machine learning model and user interface on HuggingFace Spaces offers numerous benefits in terms of accessibility, ease of use, and collaboration. Here is an explanation of the process and the advantages of deploying on HuggingFace Spaces:

1. **Deploying the Model**
   - HuggingFace Spaces: HuggingFace Spaces is a platform that allows developers and researchers to showcase and share their trained models. It provides a user-friendly interface and infrastructure to deploy machine learning models.
   - Uploading the Model: To deploy the model on HuggingFace Spaces, we upload the model files, including the trained weights and any necessary dependencies. HuggingFace Spaces provides tools and guidelines for packaging and uploading the model in the desired format.
   - Configuring the Environment: Once the model is uploaded, HuggingFace Spaces allows us to configure the runtime environment, specifying the required resources, dependencies, and execution settings. This ensures that the model runs smoothly and efficiently on the platform.
2. **Deploying the User Interface**
   - Integrating with HuggingFace Spaces: In addition to the model, we deploy the user interface designed using the Gradio library on HuggingFace Spaces. This integration enables users to interact with the recommender system directly on the platform.
   - Packaging the Interface: The user interface code, including the prediction function and interface configuration, is packaged and uploaded to HuggingFace Spaces. We ensure that all necessary dependencies and requirements are included in the package.
   - Connecting the Model and Interface: HuggingFace Spaces provides mechanisms to connect the deployed model with the user interface. This ensures that the interface can make requests to the model, retrieve predictions, and display the results seamlessly.
3. **Benefits of HuggingFace Spaces**
   - Accessibility: Deploying on HuggingFace Spaces makes the model and interface readily accessible to a wide range of users. They can interact with the recommender system without the need for local installations or complex setup procedures.

- Collaboration and Sharing: HuggingFace Spaces fosters collaboration and knowledge sharing within the machine learning community. Researchers and developers can easily showcase their work, allowing others to explore, evaluate, and benefit from their models and interfaces.
- Reproducibility: HuggingFace Spaces ensures reproducibility by providing a centralized platform for model deployment. Other researchers can access the deployed model, interface, and associated code, enabling them to reproduce experiments, build upon existing work, and validate the results.

Deploying the final model and interface on HuggingFace Spaces ensures accessibility, collaboration, and reproducibility, allowing others to easily interact with and benefit from our work. The platform's capabilities and documentation provide guidance and resources for successful model deployment, facilitating the dissemination and utilization of our movie recommender system.

6 Additional Tasks

1. Search Function: We implemented a search function to handle incorrect input movie names. This function allows users to obtain results even if they make mistakes or misspell the movie titles. It improves the user experience by providing more flexibility and robustness in the input handling process.

2. Content-Based Method: In addition to the cluster-based method, we implemented a content-based filtering approach. This method leverages the textual features of the movies, such as genres and titles, to generate recommendations based on similarity. By incorporating this method, you expanded the diversity and accuracy of the recommendations.

3. Collaborative-Based Method: Alongside the content-based filtering, we also implemented a collaborative filtering method. This approach recommends movies based on the preferences and ratings of similar users. By considering the preferences of other users with similar tastes, the collaborative-based method enhances the personalization and relevance of the recommendations.

4. UI Theme Enhancement: We improved the user interface by adding a theme to the Gradio model. This customization enhances the visual appeal and aesthetics of the interface, creating a more engaging and immersive experience for the users.

5. Examples in UI: To assist users and provide guidance on how to input movie titles and select the recommendation method, We included examples in the Gradio interface. These examples demonstrate the proper format and showcase the functionality of the recommender system, improving the user experience and making it easier for users to interact with the system.

6. Evaluation and Ranking: After generating the recommendations, We evaluated the results of each method and determined their ranking. Based on the evaluation, We found that the content-based method performed the best, followed by collaborative filtering, and then cluster-based filtering. We incorporated this ranking information into the description in the Gradio interface, providing users with insights into the performance of each method.

7. Data Preprocessing: We took extra care in preprocessing the data to minimize data loss. By applying appropriate techniques and handling missing values, outliers, and other data quality issues, you ensured that the dataset was clean and ready for analysis. This preprocessing step contributes to the accuracy and reliability of the recommender system.

8. Clean Code: We prioritized writing clean and readable code that is easy to understand, modify, and maintain. Clean code improves the overall project quality, facilitates collaboration, and allows for future enhancements and optimizations.

By completing these additional tasks, We enhanced the functionality, performance, and user experience of the movie recommender system. These improvements demonstrate your dedication to delivering a comprehensive and impactful project.

Links

- HuggingFace link:
  https://huggingface.co/spaces/Mahziar/Mahziar-Recommender-System
  or click Here

- Dataset link:
  https://www.kaggle.com/datasets/rounakbanik/the-movies-dataset
  or click here