Movie Recommendation System

Pitta Manoj
Electronics and Communication Engineering
National Institute of Technology Karnataka
Surathkal, India
manojpitta@gmail.com

Pratheek KB
Electrical and Electronics Engineering
National Institute of Technology Karnataka
Surathkal, India
pratheekkb93@gmail.com

Siddh Shetty
Electrical and Electronics Engineering
National Institute of Technology Karnataka
Surathkal, India
shettysiddh23@gmail.com

Nischith Gowd
Mechanical Engineering
National Institute of Technology Karnataka
Surathkal, India
nischith0708@gmail.com

***Abstract*—As the demand for business solutions intensifies, there is a growing reliance on extracting meaningful insights from vast volumes of raw data. This trend is particularly notable in digital recommendation systems, which have become integral in consumer industries spanning books, music, clothing, movies, news articles, places, utilities, and more. These systems leverage user data to enhance future suggestions. This paper seeks to outline the implementation of a movie recommendation system employing two movie recommender based algorithms such as content based and collaborative filtering. Additionally, the study delves into data analysis to uncover insights within the movie dataset, utilizing scikit-learn (often referred to as sklearn) libraries in Python.**

Keywords: Collaborative Filtering, Movie Lens Data set , SVD, TF-IDF, NLP, Tinkter

I. Introduction

A recommendation system, also known as a recommendation engine, serves as a model employed in information filtering. Its primary function is to anticipate a user's preferences and offer suggestions based on these preferences. These systems have gained significant popularity and are extensively utilized in diverse domains, including movies, music, books, videos, clothing, restaurants, food, and various other utilities.By gathering information about user preferences and behavior, these systems continually enhance their recommendations.

Numerous companies are leveraging recommendation systems to enhance user engagement and elevate the overall shopping experience. The advantages of recommendation systems are manifold, with customer satisfaction and revenue being paramount. Users often rely on recommendations derived from their past transactions, anticipating improved options. When these suggestions align closely with user needs, customer satisfaction is heightened, fostering repeated use of the application. The consistent use of such applications contributes substantially to revenue generation, prompting many e-commerce entities to refine and optimize their recommendation engines.

While recommendation systems are prevalent, creating systems that deliver accurate and relevant suggestions poses a considerable challenge. Each user exhibits unique preferences and likes, further influenced by factors like mood, occasion, and the purpose of their purchase. Inability to predict and provide suitable recommendations tailored to user preferences can lead to user disengagement from a website or app. Hence, there exists a continual imperative for companies to enhance and refine their recommendation systems to meet evolving user expectations.

One of the primary objectives of this paper is to develop a movie recommendation system that takes into account the historical movie ratings given by diverse users to offer personalized suggestions. This system was implemented using a hybrid approach, combining collaborative filtering and content-based algorithms, leveraging the capabilities of the scikit-learn framework in Python.

This paper is organized as follows: First, a brief overview of a few relevant, recent research done in the space of recommender system will be discussed. Second, we will present the understanding on the technique of Hybrid filtering. Finally, a user interface on the Hybrid filtering used will be presented.

II. Overview

A. Recommendation Systems

In this section, we provide a brief overview of various types of recommendation systems. There are three primary categories: collaborative filtering, content-based filtering, and hybrid systems.
Collaborative Filtering: This type of system examines user behavior and preferences to predict their preferences based on similarities with other users. Collaborative filtering includes two subtypes: user-based recommender and item-based recommender.

Content-Based Filtering: Content-based systems take into account the description and features of an item, combined with the user's preferences, to offer personalized suggestions.
Hybrid System: Hybrid recommendation systems integrate both collaborative and content-based filtering approaches.
These systems execute collaborative and content-based predictions independently, and the outcomes of both methods are amalgamated to furnish comprehensive recommendations.

B. User Interface Application

To enhance user experience and accessibility, a graphical user interface (GUI) has been implemented using the Tkinter library in Python. Tkinter provides a user-friendly environment that allows users to interact seamlessly with the recommender system.

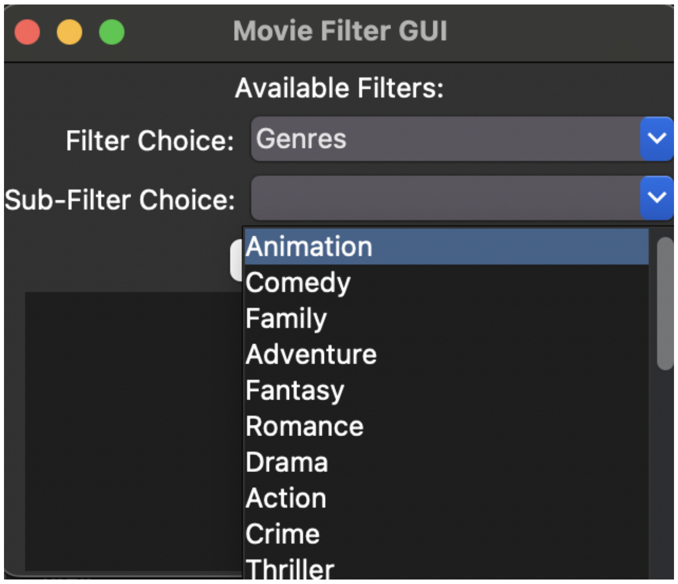


Fig. 1. Figure depicting GUI

C. Related Work

In this section, we explore related work on the model-based approach coupled with feature engineering. Model-based approaches commonly involve probabilistic methods, Bayesian networks, nearest neighbors algorithms, and bio-inspired algorithms like neural networks and genetic algorithms.

A notable method for predicting user preferences in recommendation systems is detailed by Antonio Hernando et al. This method, based on non-negative matrix factorization collaborative filtering and a Bayesian probabilistic model, competes favorably with classical matrix factorization.

Jian Wei1 et al. introduce a method grounded in deep learning neural networks, specifically addressing both the complete cold start problem (where no rating records are available) and the incomplete cold start problem (few rating records available). Notably, this method excels in addressing the complete cold start problem.

Kebin Wang and Ying Tan describe an improved naïve Bayesian algorithm aimed at enhancing recommendation quality. However, challenges arise in adhering strictly to conditional independence, particularly in calculating item independence.

Lifang Ren et.al. propose a collaborative service recommendation method based on support vector machines, effectively filtering out user-preferred services and yielding superior recommendation quality and efficiency.

Qinbao Song et.al. present a novel software process model recommendation method, leveraging various classification algorithms, including single learner and ensemble learner, for the recommendation of suitable project models.

Despite the effectiveness of model-based recommendations, particularly the naïve Bayesian approach for prediction accuracy, challenges persist, such as the cold start problem. Consequently, hybrid approaches have gained prominence to address these challenges and enhance prediction accuracy, employing weighted, switching, mixed, feature combination, cascade, feature augmentation, and meta-level techniques. This paper adopts a hybrid approach by combining naïve Bayesian with feature engineering.

The literature on feature engineering is then explored. Chun-Liang Li et al. propose a feature engineering technique for the Knowledge Discovery and Data Mining (KDD) Cup Challenge 2013, generating various features from available paper author identification information.

Jianjun Xie et. al. present a method for segregating highly rated songs from unrated songs in a large set of Yahoo Music dataset. This method generates features from the k-nearest neighbors of users and k-nearest neighbors of items, with a focus on item-based features due to the observed weakness of user-based k-nearest neighbors.

Motivated by the aforementioned literature, this paper proposes a hybrid method of recommendation.

III. Hybrid Filtering Model

We propose a hybrid movie recommendation system that focuses on the ratings given by users as well as the keywords in our dataset to provide recommendations.We utilized a hybrid model, incorporating both user-based and collaborative filtering techniques. Additionally, we integrated Singular Value Decomposition (SVD) algorithms to enhance recommendation accuracy. To make the user experience seamless, we developed a user interface to interact with the system. In this section, we describe the intricacies of the of hybrid filtering.

A. User-based filtering

In the realm of collaborative filtering for movie recommendations, we employed the Surprise library to implement a Singular Value Decomposition (SVD) algorithm. The ratings data, consisting of user ratings for movies, was loaded and split into training and testing sets. The Collaborative Filtering (CF) model was trained on the training set using the SVD algorithm.

To assess the performance of our hybrid filtering model, we evaluated its predictions on the test set, computing the Root Mean Square Error (RMSE) as a measure of accuracy. The obtained HF RMSE provides insights into how well the model aligns with the actual user ratings.

The model's functionality extends beyond evaluation, allowing us to generate personalized movie recommendations for users. For instance, we created a function,"get_cf_recommendations", which takes a user ID as input and returns the top N movie recommendations based on the user's preferences. These recommendations are determined by predicting ratings for unrated movies and selecting the highest estimated ratings.

As an illustration, we applied the collaborative filtering recommendations for a user with ID 1, showcasing a set of movies that align with the user's likely preferences. This collaborative filtering approach leverages user behavior and preferences to offer tailored movie suggestions, enhancing the overall user experience.

B. *Item-basedfiltering*

Unlike collaborative filtering, where recommendations are based on user preferences, our content-based filtering approach focuses on the unique features of each movie, particularly the keywords associated with them. To optimize our system, we take an additional step by removing unnecessary columns from the dataset during the search process.
In this content-based approach, we start by creating a dataset that includes movie titles and their keywords. We then employ the TF-IDF method to understand the importance of each keyword in describing a movie. After this initial setup, we calculate the cosine similarity between movies, measuring how alike they are based on their keywords.
The distinguishing feature of our content-based method lies in the proactive identification of the most similar movies for each one ahead of time. This preemptive calculation streamlines the recommendation process. When a user seeks suggestions, the system rapidly identifies movies with the closest keyword profiles to the user's favorites, ensuring that the recommendations feel personal and relevant.
By paying attention to the words describing each movie, our content-based method gains a deeper understanding of the themes and unique aspects of each film. Moreover, the removal of unnecessary columns during the search process optimizes the system's efficiency. This focused approach to content characteristics enhances the system's ability to cater to individual tastes, providing a personalized and streamlined movie recommendation service.

IV. Implementation

In this section, we provide a detailed description of the implementation of our movie recommendation system. The system incorporates various components and algorithms to enhance the accuracy and personalization of movie suggestions.

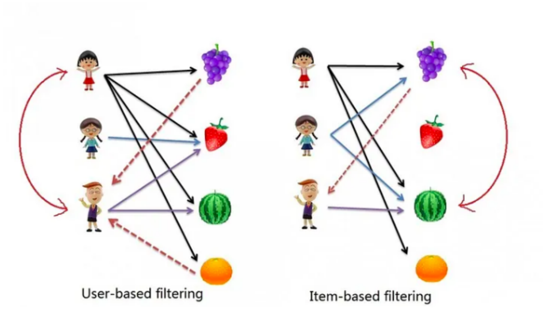


Fig. 2. Figure depicting user based and item based filtering

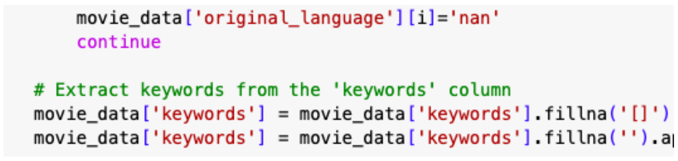
A. *Dataset*

Our movie recommendation system relies on a diverse and comprehensive dataset, comprising several key sources. Each dataset plays a crucial role in capturing different aspects of movie information and user preferences. The Movie Metadata dataset serves as the backbone of our system, providing essential information about each movie. This includes details such as movie titles, genres, release dates, and original languages. The dataset is a rich source of information that forms the basis for both collaborative and content-based filtering.Here's an overview of the data we used:

- Credits Data: The credits dataset complements our movie information by offering insights into the cast and crew involved in the production of each film. By incorporating information about actors and crew members, our system gains a deeper understanding of the movie's creative contributors.
- Keywords Data: Keywords are vital for content-based filtering. The keywords dataset contains information about the significant terms associated with each movie. We leverage this data to extract relevant features that contribute to the content-based recommendation algorithm.
- Links Data: The links dataset provides essential linkage between our movie dataset and external databases. It includes crucial identifiers such as movie IDs, allowing us to establish connections and references across different datasets seamlessly.
- Ratings Data: User preferences are a central component of our recommendation system. The ratings dataset contains user-provided ratings for various movies. This data is fundamental for collaborative filtering, enabling the system to understand user behavior and preferences.

B. *Data Cleaning*

In the data cleaning process, the initial steps involve handling missing values in the 'release_date' and 'original_language' columns of the movie dataset. For the 'release_date', missing values are replaced with 'NAN' and only the year portion is retained. Similarly, missing or invalid language entries are replaced with 'nan'. Additionally, in the 'keywords' column, missing values are filled with an empty list, and the data is transformed into a string format. The subsequent step focuses on data filtering, where a new dataset, ‘filtered_movie_data‘, is created based on user-selected filters. Movies that do not meet the specified criteria are removed from the dataset. Lastly, the TF-IDF vectorization technique from scikit-learn is applied to the ‘keywords‘ column, converting it into numerical vectors. These cleaning and preprocessing steps collectively ensure that the dataset is consistent, handles missing values appropriately, and is well-prepared for subsequent collaborative and content-based filtering analyses.



```
        movie_data['original_language'][i]='nan'
        continue

  # Extract keywords from the 'keywords' column
  movie_data['keywords'] = movie_data['keywords'].fillna('[]')
  movie_data['keywords'] = movie_data['keywords'].fillna('').a
```

Fig. 3. Figure depicting data cleaning

C. Data Analysis

Our study of the movie dataset using Python’s Matplotlib ,Seaborn and Pandas libraries, we found patterns and interesting data.
The first plot, a scatter plot, helped us see how popular movies are in the dataset. It gave a quick look at the range of popularity.

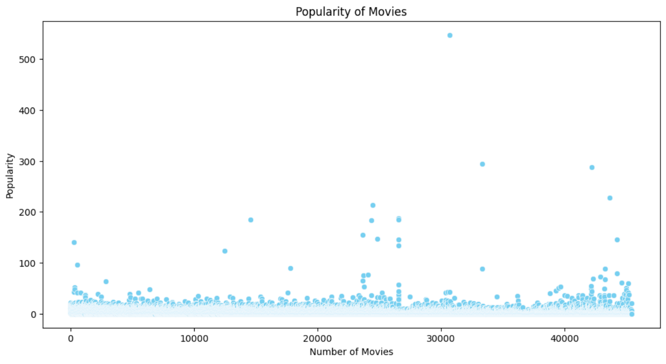


Fig. 4. Scatter plot on popularity of movies

Then, we checked out the different movie genres. Using a bar plot, we showed how many movies belong to each genre. This simple visual helped us understand which genres are more common in our dataset.
We explored how different movie features, like popularity and budget, are connected. We used a heatmap to make this easy to see. The colors on the heatmap show how strong the connections are between these features.
Incorporating additional functionality, we delved into the relationship between movie budgets and revenues. This analysis utilized Pandas to convert, filter, and transform the dataset, followed by a scatter plot that differentiated movies below and above the budget-revenue line. This visual representation, complemented by red line indicating budget equals revenue, offered insights into the financial dynamics of movies.
These plots help us understand our movie data better. They are

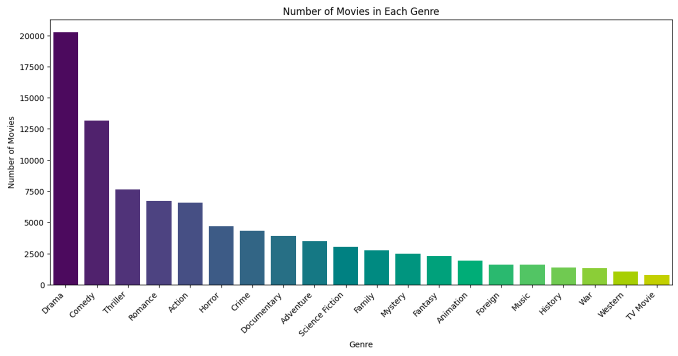


Fig. 5. Bar graph on number of movies in each genre

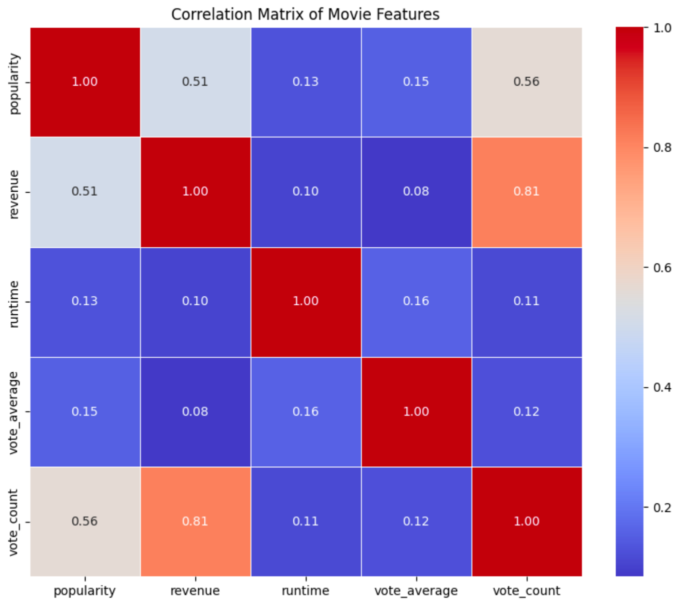


Fig. 6. Heat map of different movie features

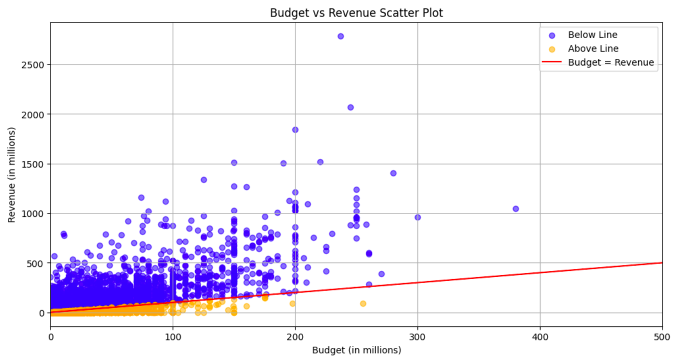


Fig. 7. Scatter plot on Budget vs Revenue

like a map guiding us to make smarter decisions in developing our system.

D. Model Building

In the process of building our movie recommendation system, we utilized collaborative filtering and content-based filtering techniques. For collaborative filtering, we employed the Surprise library to implement Singular Value Decomposition (SVD) through the SVD algorithm. This model was trained on the user-item rating data, providing us with a hybrid filtering model capable of predicting movie ratings and making personalized recommendations based on users' historical preferences.

Additionally, for content-based filtering, we incorporated a content recommendation system using the TF-IDF (Term Frequency-Inverse Document Frequency) vectorization method.

In our approach to building the movie recommendation system, we employed TF-IDF (Term Frequency-Inverse Document Frequency) as a key feature extraction technique. TF-IDF is a numerical representation that assesses the significance of words within the textual content of documents, considering both the frequency of a term in a specific document (TF) and its rarity across a collection of documents (IDF). For each movie in our dataset, we applied TF-IDF to the keywords associated with it, generating a TF-IDF vector that captures the importance of each keyword to that particular movie. This vectorization process results in a high-dimensional representation of the textual content. To measure the similarity between movies, we computed the cosine similarity between their respective TF-IDF vectors. The higher the cosine similarity, the more similar the content of two movies is considered to be. This allows us to recommend movies with similar textual features, enhancing the system's ability to provide relevant and content-specific suggestions to users.

$$TFIDF\ score\ for\ term\ i\ in\ document\ j = TF(i,j) * IDF(i)$$

$$where$$

$$IDF = Inverse\ Document\ Frequency$$

$$TF = Term\ Frequency$$

$$TF(i,j) = \frac{\text{Term i frequency in document j}}{\text{Total words in document j}}$$

$$IDF(i) = \log_2\left(\frac{\text{Total documents}}{\text{documents with term i}}\right)$$

$$and$$

$$t = Term$$

$$j = Document$$

Fig. 8. Figure depicting TF-IDF

In our approach, cosine similarity also serves as a key metric to assess the likeness between movies in terms of their keyword content. Initially, we employ TF-IDF vectorization to represent each movie's keywords numerically. Subsequently, we compute the cosine similarity between these TF-IDF vectors, allowing us to gauge the similarity between pairs of movies. When a user selects a movie, its TF-IDF vector is retrieved, and the cosine similarity is calculated against all other movies in the dataset. The resulting similarity scores aid in identifying movies with content similar to the selected one. Top recommendations are then generated by ranking movies based on their cosine similarity scores, ensuring a personalized and content-driven recommendation system.

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

Fig. 9. Cosine similarity formula

We also employ Singular Value Decomposition (SVD) as a key component in collaborative filtering for personalized movie recommendations. SVD is a matrix factorization technique that decomposes the user-movie rating matrix into three matrices: user features, singular values, and movie features. The latent factors extracted from these matrices capture hidden patterns and preferences, enhancing the model's ability to predict user ratings for unrated movies. To assess the model's performance, we utilize Root Mean Square Error (RMSE) as an evaluation metric. This metric measures the difference between the actual and predicted ratings, providing insights into the accuracy of our collaborative filtering approach. By leveraging SVD and monitoring RMSE, our model aims to deliver precise and tailored movie recommendations based on user preferences.

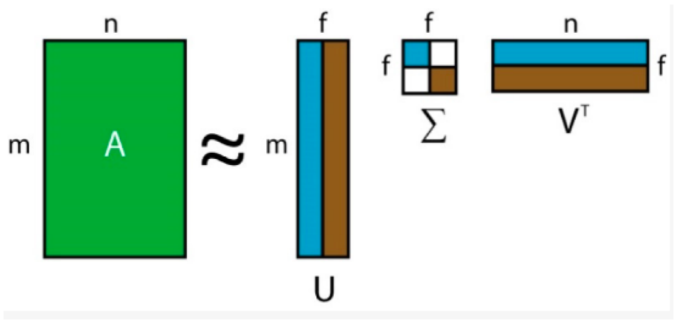
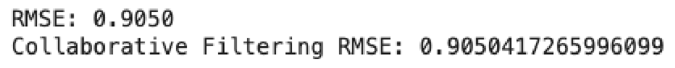


Fig. 10. Figure depicting SVD

In our collaborative filtering model,the Root Mean Squared Error (RMSE) is a performance metric that quantifies the average difference between the actual user ratings and the ratings predicted by the model. A lower RMSE indicates a better fit of the model to the observed data, reflecting the accuracy of the collaborative filtering algorithm in predicting user preferences for movies.

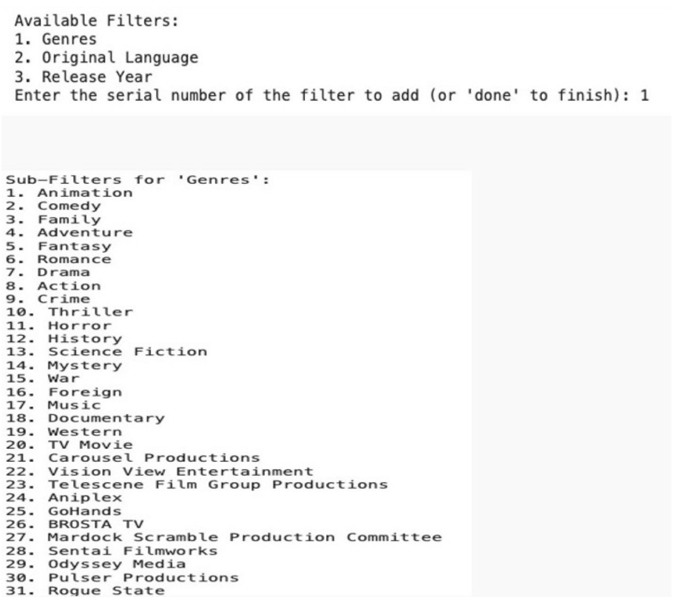


```
RMSE: 0.9050
Collaborative Filtering RMSE: 0.9050417265996099
```

Fig. 11. Figure depicting RMSE

E. Results

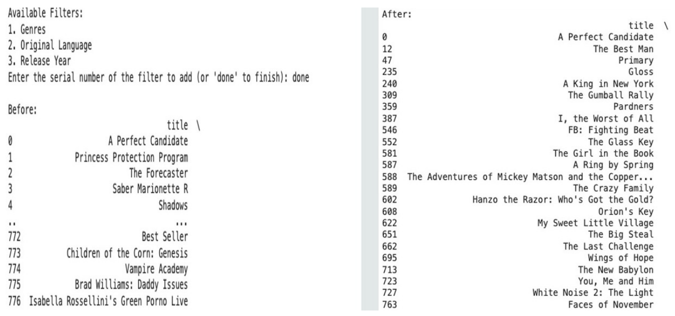
The results of our model for the implemented content based system is shown below



```
Available Filters:
1. Genres
2. Original Language
3. Release Year
Enter the serial number of the filter to add (or 'done' to finish): 1
```

```
Sub-Filters for 'Genres':
1. Animation
2. Comedy
3. Family
4. Adventure
5. Fantasy
6. Romance
7. Drama
8. Action
9. Crime
10. Thriller
11. Horror
12. History
13. Science Fiction
14. Mystery
15. War
16. Foreign
17. Music
18. Documentary
19. Western
20. TV Movie
21. Carousel Productions
22. Vision View Entertainment
23. Telescene Film Group Productions
24. Aniplex
25. GoHands
26. BROSTA TV
27. Mardock Scramble Production Committee
28. Sentai Filmworks
29. Odyssey Media
30. Pulser Productions
31. Rogue State
```

Fig. 12. Content Based Output



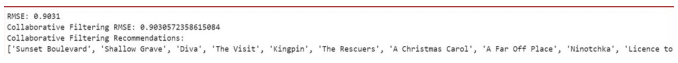
```
Available Filters:
1. Genres
2. Original Language
3. Release Year
Enter the serial number of the filter to add (or 'done' to finish): done

Before:
                                 title  \
0                    A Perfect Candidate
1             Princess Protection Program
2                         The Forecaster
3                      Saber Marionette R
4                                Shadows
..                                   ...
772                          Best Seller
773          Children of the Corn: Genesis
774                       Vampire Academy
775           Brad Williams: Daddy Issues
776  Isabella Rossellini's Green Porno Live
```

```
After:
                                                title  \
0                                 A Perfect Candidate
12                                       The Best Man
47                                            Primary
235                                             Gloss
240                                  A King in New York
309                                  The Gumball Rally
359                                          Pardners
387                               I, the Worst of All
546                                 FB: Fighting Beat
552                                     The Glass Key
581                               The Girl in the Book
587                                  A Ring by Spring
588  The Adventures of Mickey Matson and the Copper...
589                                  The Crazy Family
602                Hanzo the Razor: Who's Got the Gold?
608                                      Orion's Key
622                           My Sweet Little Village
651                                     The Big Steal
662                                The Last Challenge
695                                     Wings of Hope
713                                   The New Babylon
723                                  You, Me and Him
727                           White Noise 2: The Light
763                                 Faces of November
```

Fig. 13. Before and After applying our filter

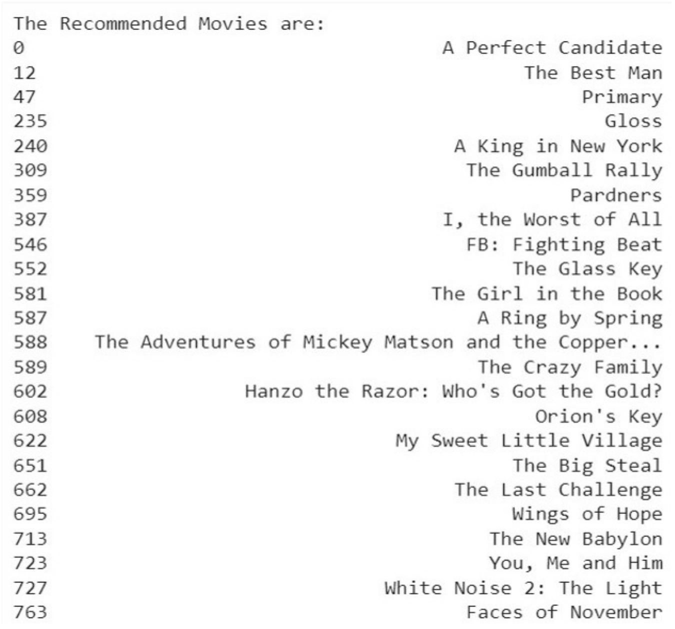
The output of the collaborative model implemented using SVD is shown below



```
RMSE: 0.9031
Collaborative Filtering RMSE: 0.9030572358615084
Collaborative Filtering Recommendations:
['Sunset Boulevard', 'Shallow Grave', 'Diva', 'The Visit', 'Kingpin', 'The Rescuers', 'A Christmas Carol', 'A Far Off Place', 'Ninotchka', 'Licence to
```

Fig. 14. Collaborative filtering Output

Combining both to form our hybrid model,the output after applying the filtering is shown below



```
The Recommended Movies are:
0                                   A Perfect Candidate
12                                         The Best Man
47                                              Primary
235                                               Gloss
240                                  A King in New York
309                                   The Gumball Rally
359                                            Pardners
387                                 I, the Worst of All
546                                   FB: Fighting Beat
552                                       The Glass Key
581                                The Girl in the Book
587                                    A Ring by Spring
588   The Adventures of Mickey Matson and the Copper...
589                                    The Crazy Family
602                Hanzo the Razor: Who's Got the Gold?
608                                         Orion's Key
622                             My Sweet Little Village
651                                       The Big Steal
662                                  The Last Challenge
695                                       Wings of Hope
713                                     The New Babylon
723                                     You, Me and Him
727                            White Noise 2: The Light
763                                   Faces of November
```

Fig. 15. Hybrid Based output

V. Model Evaluation

A. Qualitative evaluation

We examined the top movie recommendations generated by the collaborative filtering algorithm. By analyzing the relevance of these recommendations to users' tastes and preferences, we gained insights into the model's ability to understand and capture subtle nuances in movie preferences. Qualitative evaluation allows us to assess the system's effectiveness in providing personalized and meaningful movie suggestions beyond numerical metrics.

B. Quantitative evaluation

We utilized the Root Mean Squared Error (RMSE) as a numerical metric to measure the accuracy of the model in predicting user ratings. A lower RMSE indicates a closer alignment between predicted and actual ratings, reflecting improved predictive accuracy.

C. Some Common Issues

In the implemented movie recommendation system, several challenges and considerations emerge that warrant attention. One notable concern is the New User Problem, wherein users who have not provided any ratings present a challenge, commonly referred to as User Cold Start. Addressing this issue involves devising effective strategies for recommending movies to users with no prior rating history.

Another noteworthy challenge is the Item Cold Start problem, particularly when a new movie is introduced to the system without any associated ratings. The system needs to discover and recommend such new items, which prompts the exploration of approaches such as recommending movies similar to the genres of top-rated movies.

Furthermore, the system's performance and scalability depend on the chosen collaborative filtering approach. For user-based collaborative filtering, dynamic neighborhoods are formed based on new ratings from different users, which can pose computational challenges. On the other hand, item-based collaborative filtering involves static similarity computations, making offline computation feasible but requiring periodic updates.

Additionally, the system's overall effectiveness hinges on the accuracy of the collaborative filtering algorithms employed, and addressing issues related to sparse data, scalability, and the trade-off between user-based and item-based approaches contributes to a robust recommendation system.

Conclusion and Future Work

In this paper, we have implemented a movie recommendation system employing hybrid based filtering. The system is developed using a Hybrid-based filtering approach, taking into account the keywords associated with movies to offer tailored movie suggestions.
For future work, enhancing the recommender system could involve adopting an advanced hybrid filtering approach. Recent research suggests that advanced hybrid systems tend to be more effective and provide more accurate recommendations. Therefore, incorporating such advancements would be a valuable improvement. Currently, our system considers keywords for recommending movies. However, in the future, additional features such as the movie genre, directors, actors, and more could be considered to further enhance the recommendation process. Moreover, exploring a new framework called Apache Prediction 10 could be worthwhile for system development. The Apache Prediction 10 is a machine learning server utilizing the technology stack of Apache Hadoop, Apache Spark, Elastic Search, and Apache Hbase to build a Universal Recommender System. .

References

[1] Y. Ren and S. Gong, "A Collaborative Filtering Recommendation Algorithm Based on SVD Smoothing," 2009 Third International Symposium on Intelligent Information Technology Application, Nanchang, China, 2009, pp. 530-532, doi: 10.1109/IITA.2009.491.

[2] Y. Tai, Z. Sun and Z. Yao, "Content-Based Recommendation Using Machine Learning," 2021 IEEE 31st International Workshop on Machine Learning for Signal Processing (MLSP), Gold Coast, Australia, 2021, pp. 1-4, doi: 10.1109/MLSP52302.2021.9596525.

[3] V. Thannimalai and L. Zhang, "A Content Based and Collaborative Filtering Recommender System," 2021 International Conference on Machine Learning and Cybernetics (ICMLC), Adelaide, Australia, 2021, pp. 1-7, doi: 10.1109/ICMLC54886.2021.9737238.