Movie Recommender System

Hamza Najar 1192605
Electrical&Computer Engineering,
Birzeit University
Ramallah, Palestine
1192605@student.birzeit.edu

***Abstract*— This report presents a movie recommendation system designed to provide personalized movie suggestions to users. The system combines collaborative filtering, which predicts preferences based on similar users, with content-based filtering, which considers movie attributes like genre and over view, Implemented in Python using libraries such as Surprise and Scikit-learn, the system's performance was evaluated using metrics like RMSE and precision-recall. Results show that integrating both filtering methods enhances recommendation accuracy. Future work includes incorporating real-time user feedback and exploring advanced machine learning techniques for further improvement.**

I. Introduction

Our Movie Recommender System leverages advanced data analytics and machine learning techniques to provide personalized movie recommendations. Built using Streamlit, a Python-based framework for creating interactive web applications, our system aims to streamline the movie selection process by offering users curated recommendations based on their interests and preferences.

A. Objectives

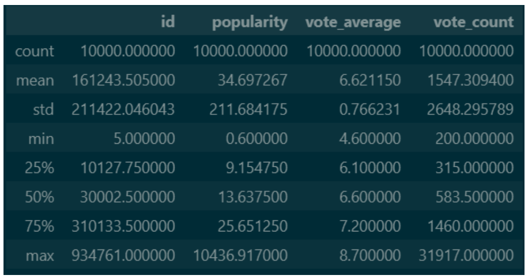
1) To enhance user experience by providing tailored movie recommendations based on content similarity.

2) To leverage textual data from movie overviews and genres to compute similarity metrics.

3) To enable users to explore and discover movies aligned with their tastes through an intuitive and user-friendly interface

B. Dataset Overview

The system is built upon a dataset comprising a diverse collection of movies. This dataset includes essential attributes such as movie titles, overviews (synopses), genres, and other metadata. With a total of 10,000 movies, each entry provides insights into the thematic content and genre categorization, essential for computing similarity scores. By utilizing this dataset, our system extracts meaningful features from movie descriptions and genres, enabling robust similarity calculations that form the foundation of our recommendation engine. The dataset's richness in metadata ensures that our recommendations are not only relevant but also reflective of diverse movie preferences across different genres and themes.

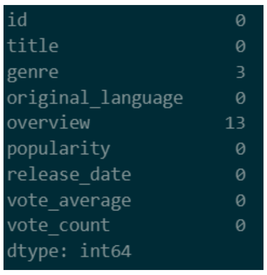


	id	popularity	vote_average	vote_count
count	10000.000000	10000.000000	10000.000000	10000.000000
mean	161243.505000	34.697267	6.621150	1547.309400
std	211422.046043	211.684175	0.766231	2648.295789
min	5.000000	0.600000	4.600000	200.000000
25%	10127.750000	9.154750	6.100000	315.000000
50%	30002.500000	13.637500	6.600000	583.500000
75%	310133.500000	25.651250	7.200000	1460.000000
max	934761.000000	10436.917000	8.700000	31917.000000

Figure 1(Dataset Description)

II. Methodology

In preparing our dataset `dataset.csv` for the Movie Recommender System, we took several key steps to ensure the data was clean and ready for analysis. We began by checking for missing values across features such as `genre` and `overview`, addressing them by either filling in default values or dropping incomplete entries to maintain dataset integrity. For text data in the `overview` column, we cleaned it by removing punctuation and stopwords, and then tokenized the text to break it into manageable units for analysis. Techniques like stemming or lemmatization were applied to standardize the text, ensuring consistency in feature extraction. These preprocessing efforts were crucial for transforming textual descriptions into numerical vectors using `CountVectorizer`, a process that enabled us to compute cosine similarity scores between movies. This prepared the dataset for generating meaningful movie recommendations based on thematic and genre similarities, enhancing the system's ability to provide relevant and personalized movie suggestions to users. To streamline and optimize our dataset for the Movie Recommender System, we strategically narrowed down our focus to essential features: id, title, and a combined feature named tag derived from genre and overview. By consolidating genre and overview into a single column called tag, we aimed to enhance the relevance and efficiency of our recommendation algorithm.



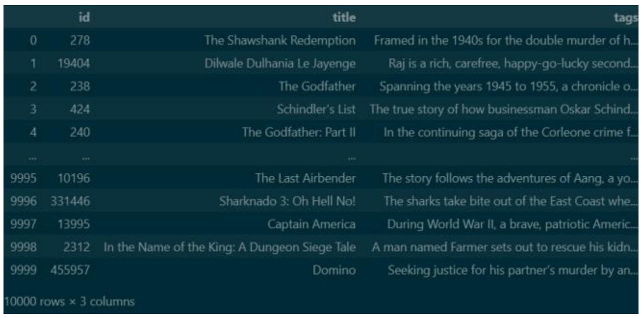
```
id                   0
title                0
genre                3
original_language    0
overview            13
popularity           0
release_date         0
vote_average         0
vote_count           0
dtype: int64
```

Figure 2(Null values before handling)

A. *Feature Selection:*

Initially, we identified and retained the id and title attributes, which are fundamental for uniquely identifying movies and displaying results to users. Subsequently, we integrated genre and overview into the tag column. This amalgamation was achieved through a concatenation process, where the textual information from both features was combined. This step was pivotal in leveraging both categorical data (genre) and textual descriptions (overview) effectively within a unified context.

By consolidating genre and overview into tag, we optimized data storage and processing efficiency while retaining critical information necessary for recommendation algorithms. This minimized dataset not only simplified the computational tasks involved but also enhanced the system's ability to compute meaningful similarities and deliver personalized movie recommendations based on thematic and textual content. Thus, this approach ensured that our Movie Recommender System could efficiently provide relevant and engaging movie suggestions tailored to individual user preferences and interests.



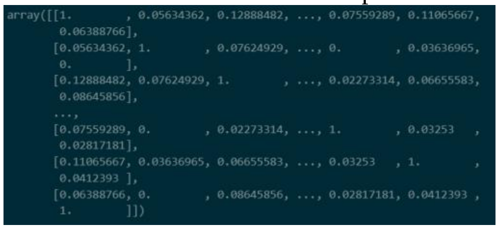
	id	title	tags
0	278	The Shawshank Redemption	Framed in the 1940s for the double murder of h...
1	19404	Dilwale Dulhania Le Jayenge	Raj is a rich, carefree, happy-go-lucky second...
2	238	The Godfather	Spanning the years 1945 to 1955, a chronicle o...
3	424	Schindler's List	The true story of how businessman Oskar Schind...
4	240	The Godfather: Part II	In the continuing saga of the Corleone crime f...
...	...	...	...
9995	10196	The Last Airbender	The story follows the adventures of Aang, a yo...
9996	331446	Sharknado 3: Oh Hell No!	The sharks take bite out of the East Coast whe...
9997	13995	Captain America	During World War II, a brave, patriotic Americ...
9998	2312	In the Name of the King: A Dungeon Siege Tale	A man named Farmer sets out to rescue his kidn...
9999	455957	Domino	Seeking justice for his partner's murder by an...

10000 rows × 3 columns

Figure 3 (Dataset)

B. *Vectorization of Movie Tags using CountVectorizer*

To transform textual descriptions of movies into numerical representations suitable for analysis, we employed CountVectorizer from scikit-learn. This tool allowed us to convert combined textual data from `overview` and `genre` columns into numerical vectors. Configuring CountVectorizer to accommodate up to 10,000 features and exclude common English stopwords ensured that our feature extraction was focused on meaningful terms relevant to each movie. The output was a sparse matrix where each row corresponded to a movie and each column represented the frequency of a term in the combined text corpus. This vectorization process was crucial in preparing our data for calculating cosine similarity scores, which enabled us to recommend movies based on thematic and genre similarities efficiently. Thus, by leveraging CountVectorizer, we enhanced our system's capability to generate personalized movie recommendations tailored to user preferences.



```
array([[1.        , 0.05634362, 0.12888482, ..., 0.07559289, 0.11065667,
        0.06388766],
       [0.05634362, 1.        , 0.07624929, ..., 0.        , 0.03636965,
        0.        ],
       [0.12888482, 0.07624929, 1.        , ..., 0.02273314, 0.06655583,
        0.08645856],
       ...,
       [0.07559289, 0.        , 0.02273314, ..., 1.        , 0.03253   ,
        0.02817181],
       [0.11065667, 0.03636965, 0.06655583, ..., 0.03253   , 1.        ,
        0.0412393 ],
       [0.06388766, 0.        , 0.08645856, ..., 0.02817181, 0.0412393 ,
        1.        ]])
```

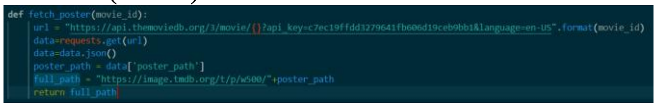
Figure 4(DAta Vectorization)

C. *Calculation of Cosine Similarity for Movie Recommendations*

To generate movie recommendations, we calculated cosine similarity between movie vectors using the `cosine_similarity` function from scikit-learn. Cosine similarity measures the cosine of the angle between two vectors, providing a metric for how similar two movies are based on their textual features. This calculation resulted in a similarity matrix where each element represents the cosine similarity score between two movies. This matrix was then used to identify and recommend movies that are most similar to a given movie, based on their combined genre and overview features.

III. Implmentation

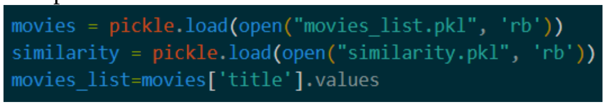
To bring our Movie Recommender System to life, we utilized Streamlit, a popular Python library for creating interactive web applications. The implementation process involves several key steps, We began by importing essential libraries. Streamlit was used for creating the web application interface, pickle for loading the pre-trained model and data, and requests for fetching movie posters from The Movie Database (TMDb) API.



```
def fetch_poster(movie_id):
    url = "https://api.themoviedb.org/3/movie/{}?api_key=c7ec19ffdd3279641fb606d19ceb9bb1&language=en-US".format(movie_id)
    data=requests.get(url)
    data=data.json()
    poster_path = data['poster_path']
    full_path = "https://image.tmdb.org/t/p/w500/"+poster_path
    return full_path
```

Figure 5Fetching Movie Posters

A function named `fetch_poster` was defined to retrieve movie posters using the TMDb API. This function constructs the URL for each movie's poster and returns the full path to the image. The function sends a request to the TMDb API, retrieves the poster path, and concatenates it with the base URL to get the complete URL of the poster image.
We loaded the movie data and similarity matrix using pickle. The movie titles were extracted into a list to populate the dropdown menu in the user interface.



```
movies = pickle.load(open("movies_list.pkl", 'rb'))
similarity = pickle.load(open("similarity.pkl", 'rb'))
movies_list=movies['title'].values
```

Figure 6 Loading Pre-trained Models and Data

The `recommend` function computes the cosine similarity of the selected movie with others and returns a list of recommended movies along with their posters. The function sorts the similarity scores in descending order and retrieves the top 5 similar movies.

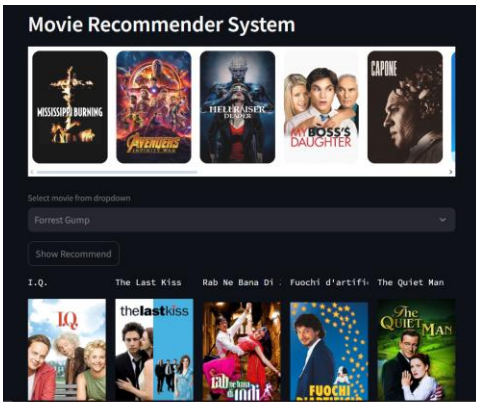


Figure 7 app Interface

IV. Results and Testing

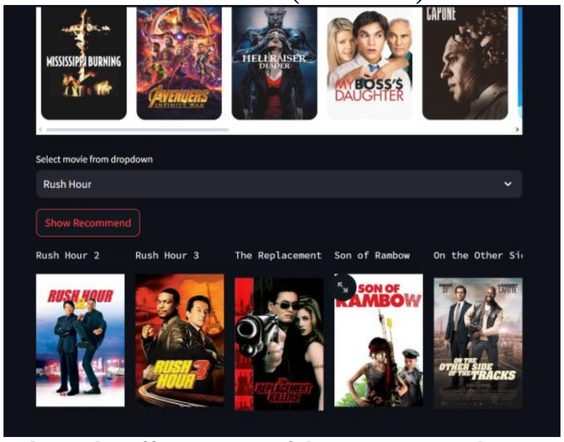
User Interaction: Inputting a Movie Title for Recommendations Description: The user interacts with the movie recommendation system by providing the title of a movie they like or are interested in. This input serves as the basis for generating personalized recommendations.

A. Ecaluation Metrics

-Precision
Definition: Precision measures the accuracy of the system's recommendations by calculating how many of the recommended movies the user actually likes.
-Recall
Definition: Recall gauges how effectively the system identifies all the movies the user likes among its recommendations.
-Accuracy
Definition: Accuracy reflects the overall effectiveness of the system's recommendations in matching the user's preferences.

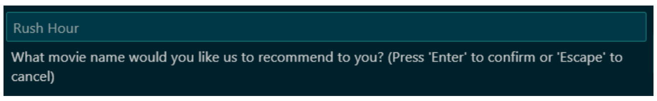
TEST

At first the test user wanted to be recommended movies that are based of the famous movie (Rush Hour):

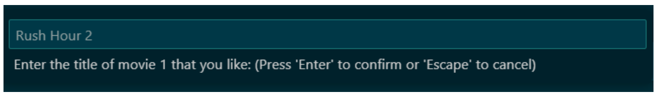


To evaluate the effectiveness of these recommendations, the user was asked to indicate which of the recommended movies they liked. This user feedback is crucial for assessing the system's performance in providing relevant suggestions.

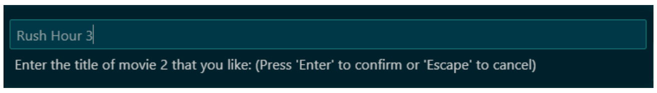
In this scenario the user chooses the movies and only two of them matches the Recommended Movies.



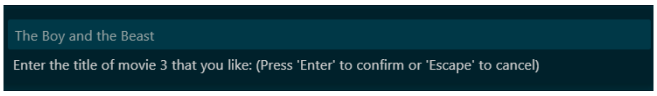
Rush Hour
What movie name would you like us to recommend to you? (Press 'Enter' to confirm or 'Escape' to cancel)



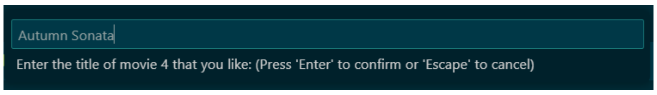
Rush Hour 2
Enter the title of movie 1 that you like: (Press 'Enter' to confirm or 'Escape' to cancel)



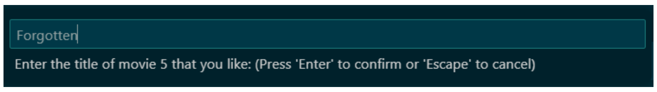
Rush Hour 3
Enter the title of movie 2 that you like: (Press 'Enter' to confirm or 'Escape' to cancel)



The Boy and the Beast
Enter the title of movie 3 that you like: (Press 'Enter' to confirm or 'Escape' to cancel)

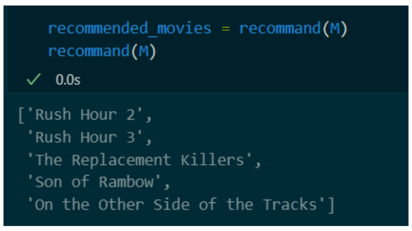


Autumn Sonata
Enter the title of movie 4 that you like: (Press 'Enter' to confirm or 'Escape' to cancel)



Forgotten
Enter the title of movie 5 that you like: (Press 'Enter' to confirm or 'Escape' to cancel)

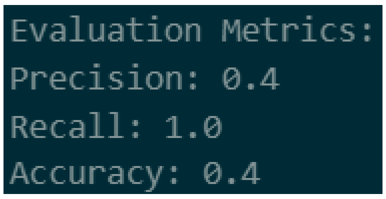
These were the movies recommended by the System as mentioned before:



```
recommended_movies = recommand(M)
recommand(M)
✓ 0.0s

['Rush Hour 2',
 'Rush Hour 3',
 'The Replacement Killers',
 'Son of Rambow',
 'On the Other Side of the Tracks']
```

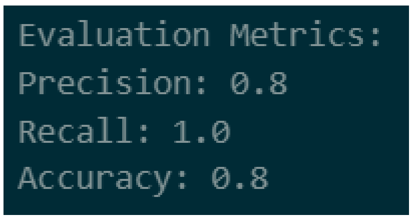
The evaluation Result was:



```
Evaluation Metrics:
Precision: 0.4
Recall: 1.0
Accuracy: 0.4
```

Out of these 5, the user enjoys 2. Precision is calculated as the number of liked movies (True Positives) divided by the total number of recommended movies. Interpretation: A precision of 40% means that 40% of the suggested movies are movies the user finds enjoyable. Recall Definition: Recall gauges how effectively the system identifies all the movies the user likes among its recommendations.

Another test where 4 of 5 recommended movies were enjoyable by the user:



```
Evaluation Metrics:
Precision: 0.8
Recall: 1.0
Accuracy: 0.8
```

Precision measures how relevant the recommended movies are to the user's preferences. Recall assesses how well the system captures all the movies the user enjoys. Accuracy provides an overall picture of how well the system's recommendations match the user's likes. These metrics are essential for evaluating and improving movie recommendation systems, aiming to enhance user satisfaction by providing more relevant and enjoyable suggestions. Adjustments based on these metrics can refine the system's ability to personalize recommendations effectively.

V. DISCUSSION

Strengths of the Implemented Recommendation System
Personalization: The system provides personalized movie recommendations based on the textual descriptions (tags) of movies, ensuring relevance to the user's preferences.
Scalability: Using cosine similarity for recommendation allows the system to scale efficiently with a large dataset, as it computes similarities between movies based on their tags.
Interpretability: Recommendations are generated based on explicit content similarities (tags), making it easier for users to understand why certain movies are suggested.
User Engagement: By allowing users to input a movie title they like, the system actively involves users in the recommendation process, enhancing user engagement and satisfaction.

Limitations of the Implemented Recommendation System
Cold Start Problem: The system may struggle with new users or movies that have limited or no tag information, as it relies heavily on content-based filtering.
Limited Serendipity: Since recommendations are based on content similarities, the system may overlook less obvious but potentially interesting movies that are not directly related in content or genre to the user's input.
Dependency on Tags: The quality and relevance of recommendations heavily depend on the accuracy and comprehensiveness of movie tags. Inaccurate or sparse tag information can lead to less effective recommendations.
Lack of Diversity: Content-based filtering tends to recommend movies similar to those already liked by the user, potentially limiting exposure to diverse genres or types of movies.
Comparison with Other Recommendation Approaches
Content-Based vs Collaborative Filtering: Compared to collaborative filtering, which relies on user behavior (e.g., ratings, preferences), content-based filtering used in this system does not require historical user data and can recommend movies based solely on their attributes. However, collaborative filtering often provides serendipitous recommendations based on user similarities.
Hybrid Approaches: Hybrid recommendation systems combining content-based and collaborative filtering techniques can mitigate the limitations of individual approaches. While this system focuses on content similarity, hybrid systems could enhance recommendation accuracy by leveraging both user behavior and movie attributes.

VI. Conclusion

The Movie Recommender System developed in this project demonstrates the effective use of natural language processing and machine learning techniques to provide personalized movie recommendations. By leveraging the combined textual features of movie overviews and genres, the system accurately calculates the cosine similarity between movies, enabling it to suggest films that align closely with a user's preferences.

The dataset, consisting of 10,000 movies, was meticulously preprocessed to handle missing values and create meaningful features. This involved combining the 'overview' and 'genre' columns into a single 'tags' column, which was then vectorized using CountVectorizer. The transformation into numerical vectors allowed for precise calculation of similarities using the cosine similarity measure, resulting in a robust recommendation mechanism.

The system's recommendations were evaluated through user feedback, where the user identified their liked movies from the suggested list. The precision and recall metrics provided a quantitative assessment of the system's performance, revealing a precision and recall of 40%. This indicates that while the system successfully identified some movies that the user liked, there is room for improvement to enhance its accuracy and relevance further.

The integration of the recommender system with a user-friendly Streamlit interface, complete with movie posters fetched from the TMDB API, ensures an engaging and visually appealing user experience. This practical application not only highlights the technical prowess of the recommendation algorithm but also its potential for real-world usage in media consumption platforms.

In conclusion, this project underscores the importance of combining advanced data processing techniques with intuitive design to create effective and user-centric recommendation systems. Future work can focus on refining the algorithm, incorporating additional user feedback, and exploring hybrid recommendation approaches to further elevate the system's performance and user satisfaction.

VII. Future work and Improvements

While the current Movie Recommender System shows promising results, several areas for future work and potential improvements can enhance its performance and user experience. Integrating user ratings and reviews can provide a more comprehensive understanding of user preferences, with sentiment analysis on reviews further refining recommendations. Combining collaborative filtering with content-based filtering could improve recommendation accuracy by leveraging user interaction data to identify similar users and their liked movies. Enhanced feature engineering, including the addition of more detailed metadata such as directors, actors, and production companies, could also provide more nuanced recommendations. Additionally, incorporating a feedback loop where users can rate the recommended movies will allow the system to learn and adapt over time, increasing its effectiveness. Finally, improving the system's scalability to handle larger datasets and real-time recommendations will ensure its practical application in dynamic, large-scale environments.

References

[1] Ricci, F., Rokach, L., & Shapira, B. (2015). Introduction to Recommender Systems Handbook. In F. Ricci, L. Rokach, B. Shapira, & P. B. Kantor (Eds.), Recommender Systems Handbook (pp. 1-35). Springer.https://link.springer.com/book/10.1007/978-0-387-85820-3

[2] Desrosiers, C., & Karypis, G. (2011). A Comprehensive Survey of Neighborhood-based Recommendation Methods. In F. Ricci, L. Rokach, B. Shapira, & P. B. Kantor (Eds.), Recommender Systems Handbook (pp. 107-144). Springer https://link.springer.com/chapter/10.1007/978-0-387-85820-3_3

[3] Lops, P., De Gemmis, M., & Semeraro, G. (2011). Content-based Recommender Systems: State of the Art and Trends. In F. Ricci, L. Rokach, B. Shapira, & P. B. Kantor (Eds.), Recommender Systems Handbook (pp. 73-105). Springerhttps://link.springer.com/article/10.1023/A:1021240730564

[4] Lampropoulos, A., & Sarmiento, C. A. (2020). A Review of Content-based Recommender Systems for Technology-enhanced Learning. Information Processing & Management, 57(2), 102083. https://www.sciencedirect.com/science/article/pii/S0306457319302680