Hadoop-Based Movie Recommendation System

Vikram Singh
Deptartment of Physics IIT Guwahati
Assam, India
vikram200121063@iitg.ac.in

Apurv Kushwaha
Deptartment of Physics IIT Guwahati
Assam, India
k.apurv@iitg.ac.in

***Abstract*—The rapid proliferation of digital content necessitates advanced recommendation systems to assist users in navigating vast information spaces. This report introduces a Hadoop-based movie recommendation system designed to enhance user experience through personalized suggestions. Leveraging the distributed computing capabilities of Hadoop, our system processes extensive datasets efficiently. The recommendation algorithms encompass content-based, collaborative filtering, and hybrid models, offering diverse and tailored movie suggestions.**

The objectives include the development of a scalable system capable of handling substantial user and movie databases. The study conducts a comparative analysis of the recommendation algorithms, evaluating their pros and cons to identify the most effective approach. The MapReduce framework in Hadoop facilitates robust data processing from user reviews, addressing challenges associated with large datasets. Practical applications in online platforms underscore the significance of personalized recommendations, motivating our research to extend recommender systems into new and challenging domains.

The anticipated outcomes encompass a fully functional Hadoop-based movie recommendation system, insights into algorithmic efficiency, scalability considerations, and a meaningful contribution to the broader field of recommendation systems. This report serves as a comprehensive exploration of the intersection between Hadoop's distributed computing capabilities and the evolving landscape of personalized content suggestions.

Click here to access the video.

I. Introduction

In the dynamic landscape of digital content consumption, recommender systems play a pivotal role in guiding users through the vast array of options available. A recommender system can be defined as any mechanism that tailors individualized recommendations, providing users with personalized guidance amid a multitude of potential choices. This report embarks on an exploration of a Hadoop-based movie recommendation system, a sophisticated application of recommender systems in the realm of personalized content suggestions.

Recommender systems, a subset of information filtering (IF) techniques, focus on presenting users with items — be it movies, music, books, news, images, or webpages — that are likely to capture their interest. The fundamental principle involves comparing a user's profile with reference characteristics and predicting the user's potential rating for items they haven't yet considered.

The multifaceted landscape of recommender systems unfolds through various approaches, broadly categorized into Collaborative Filtering (CF), Content-Based Recommending, and Hybrid Approaches. In Collaborative Filtering, recommendations are made based on the collective past ratings of all users, fostering a sense of collective intelligence. On the other hand, Content-Based Recommending suggests items similar in content to those previously liked by the user or matched to user attributes. The synthesis of both approaches into Hybrid Models harnesses the strengths of collaborative and content-based methods, providing a more nuanced and effective recommendation mechanism.

As we delve into this exploration, our aim is not only to develop a functional Hadoop-based movie recommendation system but also to conduct a comparative study among the diverse approaches prevalent in the field of recommender systems. By understanding the intricacies of Collaborative Filtering, Content-Based Recommending, and Hybrid Approaches, we strive to unravel the pros and cons of each model. In essence, this report unfolds as a journey through the evolving landscape of recommender systems, offering insights into personalized content recommendations that cater to the diverse

preferences of users.

II. Problem Statement

In today's vast digital landscape, users grapple with information overload, seeking personalized recommendations to navigate the plethora of options available. This project addresses the critical need for an efficient Hadoop-based movie recommendation system. Our goal is to implement content-based, collaborative filtering, and hybrid algorithms, leveraging Hadoop's MapReduce for streamlined data processing from user reviews. Challenges include scalability, algorithmic efficiency, and conducting a comparative study of recommendation models. The project aims to deliver a responsive system catering to diverse preferences and contribute valuable insights to the recommendation systems field.

Why It's Important

Creating an efficient movie recommendation system is not only beneficial for users seeking tailored content recommendations but also carries substantial opportunities for the entertainment industry. An effectively implemented recommendation system can boost user engagement, content consumption, and ultimately revenue.

Through the integration of content-based, collaborative filtering, and hybrid recommendation methodologies within the Hadoop framework, our project tackles the complexities of big data and personalization within the movie recommendation sector. Our goal is to develop a resilient and scalable solution that enables users to uncover movies aligning with their preferences while also offering valuable insights to content providers and distributors.

III. Architectural Details

In this paper we have proposed to implement a movie recommendation system hadoop framework. It takes the ratings, genre, actors etc. for various movies to provide movie suggestions. Our system considers the user ratings to recommend movies.

Dataset Used

We selected the MovieLens 100k dataset for our experimentation due to its wide popularity and credibility as a data source. This dataset is made available by the esteemed University of Minnesota, instilling trust in its reliability. It provides comprehensive information that aligns with the requirements of our content-based analysis. Specifically, the dataset encompasses a collection of 100,000 ratings contributed by 943 users, covering a diverse array of 1,682 movies. Notably, each user has rated a minimum of 20 movies, ensuring a robust dataset. Moreover, the dataset includes detailed movie information, such as titles, cast members, release dates, and genres, which further enriches our analysis.

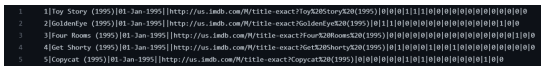


Fig. 1. | movie id | title | release date | IMDb URL | (GENRE) |

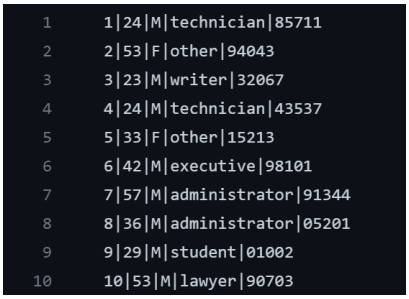


Fig. 2. user id | age | gender | occupation | zip code

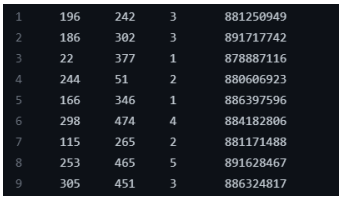


Fig. 3. | User ID | Movie ID | Rating | TimeStamp |

Models

Content-Based Model

A pure content-based recommender system only uses the profile it has created from studying the content of the things a user has previously rated to provide recommendations to that user. Based on an item's content information, this approach makes recommendations to users. For instance, in our scenario, we can recommend films based on genres that are comparable to the user's preferred

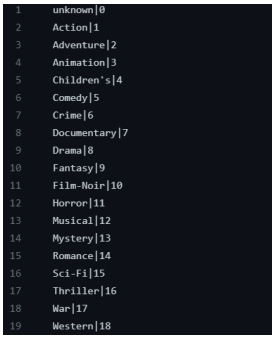


Fig. 4. | Genre | ID |

genre. To make recommendations to users, we make use of both the Item Profile and User Profile.

1) Item Profile: The Item Profile is represented as a vector of size 18 (the number of genres), with 1 or 0 marking the presence of a genre.

2) User Profile: To create a User Profile, follow these steps:

1) Find the average rating given by each user.
2) Subtract this average from all ratings of a user to get Normalized Ratings.
3) Suppose a user rated 'n' movies with 'A' genre, then the profile weight of genre 'A' for that user will be calculated as:

$$U_A = \frac{r_1 + r_2 + r_3 + \ldots + r_n}{n}$$

Where $r_1, r_2, r_3, \ldots, r_n$ are the Normalized Ratings for that user.

Similarly find U_B, U_C ... to get the User Profile for that user as:

$$U = \begin{bmatrix} U_A & U_B & U_C & \ldots & 18 \end{bmatrix}$$

3) Predicting: To offer recommendations to a user, we employ Cosine Similarity to compute a similarity metric for all the items the user hasn't yet rated. We then suggest the top 'h' items that exhibit a strong similarity measure to the user.

Cosine Similarity Calculation

The similarity parameter (Cosine Similarity) is calculated as follows:

$$\text{Similarity Parameter} = \cos(\theta) = \frac{U \cdot I}{|U|^*|I|}$$

Where:
- U represents the User Profile.
- $|U|$ is the magnitude of vector U.
- $|I|$ represent the Item Profile.

This formula measures the cosine similarity between the User Profile U and a Item Profile$I$ where $U \cdot I$ is a dot product.

Collaborative Filtering Model

Collaborative Filtering operates by predicting how a user will feel about various items and suggesting the most suitable items based on their historical preferences and the preferences of other users who share similar tastes. It relies on the idea that users who have previously shown agreement in their assessments of particular items are likely to continue agreeing in the future. Think of it as seeking movie recommendations from friends who have comparable tastes. Collaborative filtering systems are designed to deliver personalized recommendations by analyzing user behavior and preferences. In this scenario, we will employ an item-item collaborative filtering model.

Consider there are 'U' users and 'I' items.

1) Let V_i for item i be a vector of size U where the values represent ratings given by users (and 0 if no rating is given).

2) Calculate the average rating for each item's vector V_i, excluding ratings with a value of 0. For example, if 'n' users gave ratings to item 'i', sum all the ratings and divide the sum by 'n' to get the average.

3) Subtract this average from all non-zero values in the vector V_i to obtain the normalized vector.

4) Repeat the above steps to calculate the normalized vector for each item.

5) To calculate the similarity between items, we compare their normalized vectors. Let V_a and V_b be the normalized vectors of items 'a' and 'b', respectively. The similarity between these items can be calculated using the cosine similarity formula:

$$\text{Sim}(a, b) = \frac{V_a \cdot V_b}{|V_a|^*|V_b|}$$

Where:
- V_a represents the normalized vector of item 'a'.
- V_b represents the normalized vector of item 'b'.
- $\cdot$ denotes the dot product of vectors V_a and V_b.

- $|V_a|$ and $|V_b|$ denote the magnitudes of vectors V_a and V_b, respectively.

6) After calculating the similarity of an item 'i' with all the other items, select the top 'N' items similar to 'i', i.e., those with a high value of Sim(a, b). These are the 'N' nearest neighbors to item 'i'.

7) After finding the 'N' nearest neighbors for all the items, the predicted rating of User 'x' for item 'i' is given by the formula:

$$R(x,i) = \frac{\sum_{j \in N(i,x)} \text{Sim}(i,j) \cdot R(x,j)}{\sum_{j \in N(i,x)} \text{Sim}(i,j)}$$

Where:
- $R(x, j)$ is the rating given by User 'x' to item 'j'.
- $N(i, x)$ is the set of items rated by User 'x' similar to item 'i', i.e., the set of 'N' neighbors which were found earlier.
- Sim(i, j) is the similarity between item 'i' and item 'j'.

Hybrid Model

In our effort to explore the potential benefits of hybrid models, we combined the techniques of a content-based model (Baseline Predictor) and a collaborative filtering model. Our hybrid model estimates a baseline prediction for the unknown rating of User 'u' for item 'i'.

The baseline prediction (b_{ui}) is calculated as follows:

$$b_{ui} = b_i + b_u + \mu$$

Where:
- b_i is the item bias
- b_u is the user bias.
- μ is the average rating of the dataset.

The item bias (b_i) and user bias (b_u) are determined by calculating the deviation of item 'i' and user 'u' from the average item and user ratings in the complete dataset.

For item bias (b_i):

$$b_i = \text{rating deviation of item } 'i'$$
$$= (\text{avg. rating for item 'i'}) - \mu$$

For user bias (b_u):

$$b_u = \text{rating deviation of user } 'u' = \frac{1}{\langle \text{avg. rating for user } 'u' \rangle} - \mu$$

Then, the predicted rating for User 'x' of item 'i' is calculated using the following formula:

$$R(x,i) = b_{xi} + \frac{\sum_{j \in N(i,x)} \text{Sim}(i,j)^* (R(x,j) - b_{xj})}{\sum_{j \in N(i,x)} \text{Sim}(i,j)}$$

Where:
- $R(x, i)$ is the predicted rating for User 'x' of item 'i'.
- b_{xi} is the baseline prediction for User 'x' of item 'i'.
- $\sum_{j \in N(i,x)}$ represents the sum over all items j in the neighborhood $N(i, x)$.
- Sim(i, j) is the similarity between item 'i' and item 'j'.

IV. Methodology and Demo

The project's workflow consists of 7 main steps, designed with the industry environment in mind. Before delving into the details, adhere to the following directory structure to maintain clarity.

HDFS Directory Structure:

A) Directories to Create:

- `/RcomSys`
  - `/OriginalData` (Contains `u.data` file, representing User Item Rating)
  - `/Items` (Contains `u.item`, providing Item details)
  - `/User` (Contains `u.user`, offering User details)
  - `/Genre` (Contains `u.genre`, presenting Genre details)

B) Automatically Created Directories:

- `/RcomSys`
  - `/UserItemRating` (Output of MapReduce Job, Modified Dataset)
  - `/ContentOut` (Output of `ContentBased.pig`)
  - `/CollabOut` (Output of `Collaborative.pig`)
  - `/HybridOut` (Output of `Hybrid.pig`)

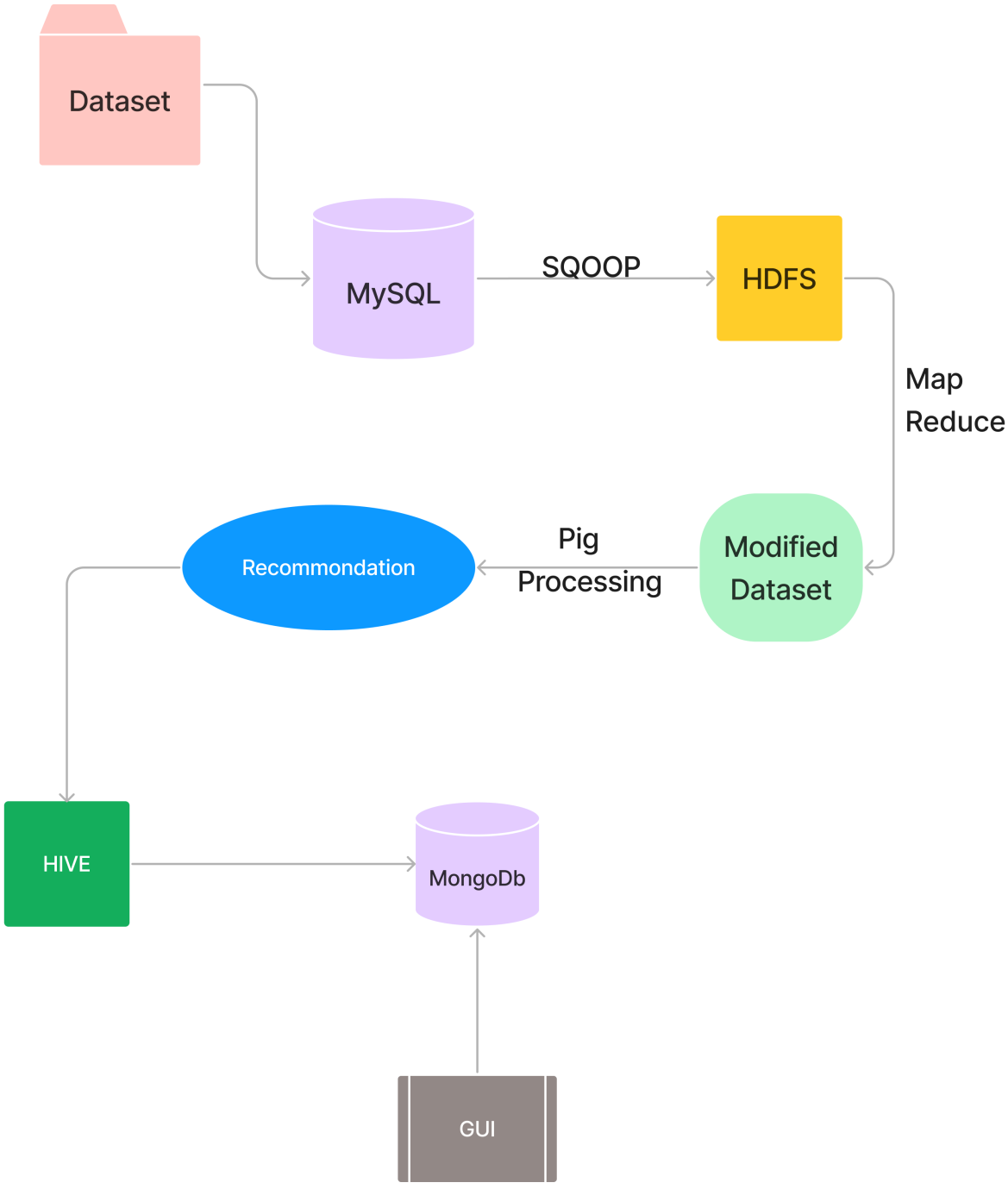


Fig. 5. Project Workflow

STEPS

Step 1: Setting Up the Environment

Components Used: MySQL

Assuming customer data is stored in a MySQL Server, we replicated the data onto our MySQL Server for further processing.

Step 2: Data Transfer to HDFS using SQOOP

Components Used: SQOOP

To facilitate Hadoop-based processing, data from MySQL Server was transferred to HDFS using SQOOP. A single command sufficed for copying data from MySQL Server to HDFS.

```
sqoop import --connect jdbc:mysql://localhost
    /<mysqlDbName> --table <mysqlTableName> --
    m1 --fields-terminated-by '\t' --target-
    dir /RcomSys/OriginalData;
```

Step 3: Data Modification

Components Used: Map Reduce

The original data lacked ratings for movies not viewed by users. In this step, we assigned a rating of 0 to unrated movies. The Map Reduce job's code (Driver, Mapper, and Reducer Classes) is attached.

```
hadoop jar DSModification.jar PDriver /RcomSys
    /OriginalData/u.data /RcomSys/
    UserItemRating
```

Step 4: Data Processing

Components Used: Pig

This crucial step involved implementing algorithms using three models for generating movie recommendations. Separate Pig Scripts for each model processed the modified data from HDFS, generating recommendations for 948 users. The Pig Scripts are attached.

```
pig ContentBased.pig
pig Collaborative.pig
pig Hybrid.pig
```

Step 5: Further Processing/Querying Output

Components Used: Hive

To gain insights into movie recommendations (5 per user), we utilized Hive for querying the output. We organized Step-4 output in Hive Tables, facilitating queries for information such as a user's preferred movie genre. The Hive Script (`hivescript1.sql`) creates the necessary tables.

```
hive -f hivescript1.sql
```

Step 6: Data Export to MongoDB

Components Used: MongoDB and Hive

To leverage MongoDB's NoSQL properties, we exported data from Hive tables to MongoDB collections. Running the Hive Script (`hivescript2.sql`) achieved this.

```
hive -f hivescript2.sql
```

Step 7: Creating GUI

Components Used: HTML, CSS, JavaScript

To visually represent the industrial environment and showcase our output, we designed a graphical user interface (GUI) using HTML for the structure, CSS for styling, and JavaScript for dynamic interactions.

Screenshots from the web application are included.

V. Results and Conclusion

Results:

To verify the accuracy of our recommendation models, we removed 100 ratings from the original dataset and predicted these omitted ratings using Collaborative Filtering and Hybrid models. The Root Mean Squared Error (RMSE) was then calculated to compare the predicted ratings with the original ratings. The results are as follows:

- **RMSE for Collaborative Filtering:** 1.004
- **RMSE for Hybrid Model:** 1.001

Conclusion:

After individually studying all three models, we observed distinct strengths within each approach:

- The **Hybrid Model** demonstrated superior performance as the best recommender among the three methods used, achieving an average RMSE of 1.0013.
- **Collaborative Filtering**, with an average RMSE of 1.0038, provided satisfactory performance and exhibited limited personalization.
- The **Content-Based Model**, concluding with an average RMSE of 0.9999999999, ranked as the least effective among the three models.

These findings highlight the nuanced strengths and weaknesses inherent in each recommendation model, with the Hybrid Model emerging as the optimal choice in this comparative analysis.

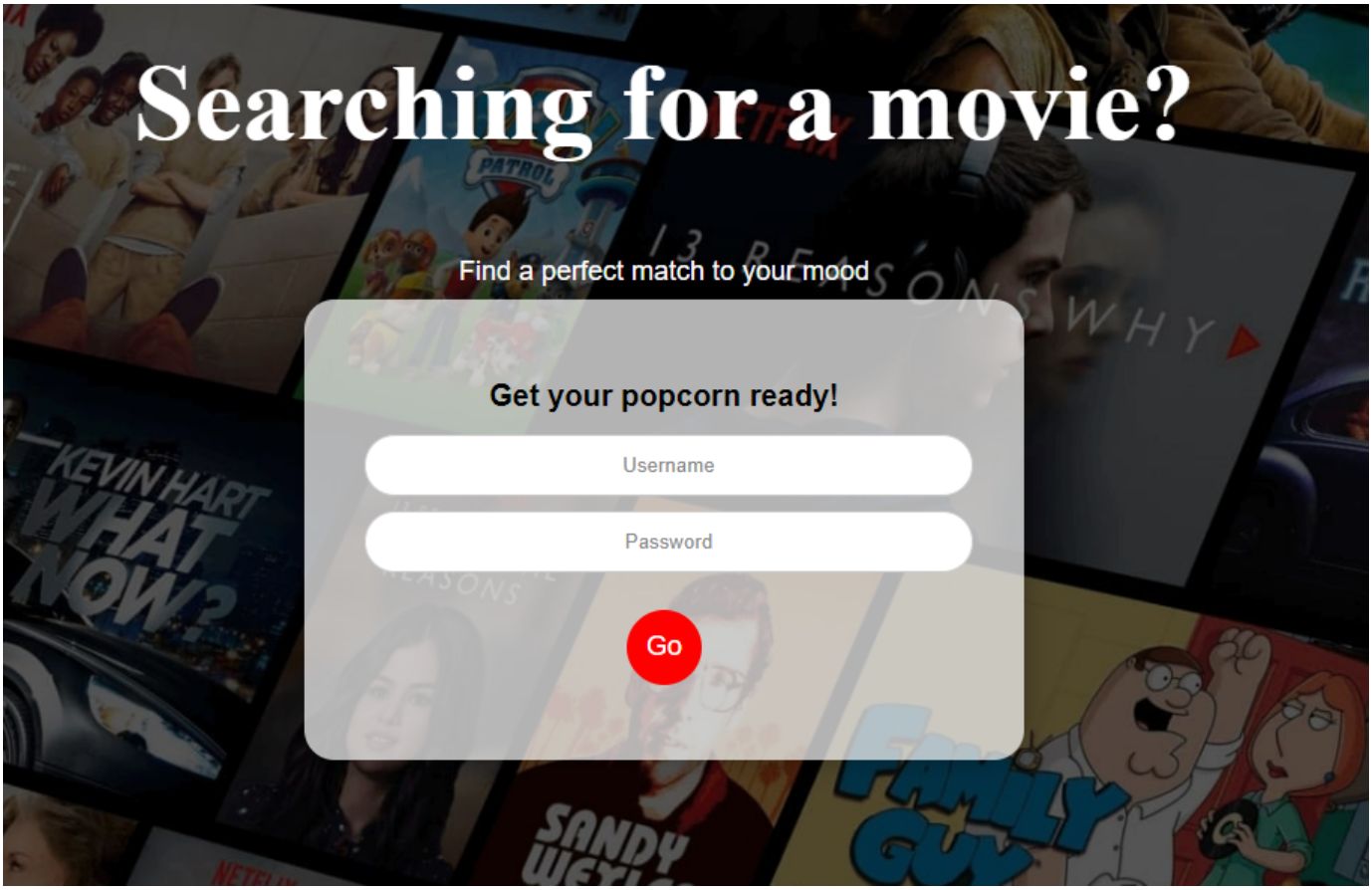
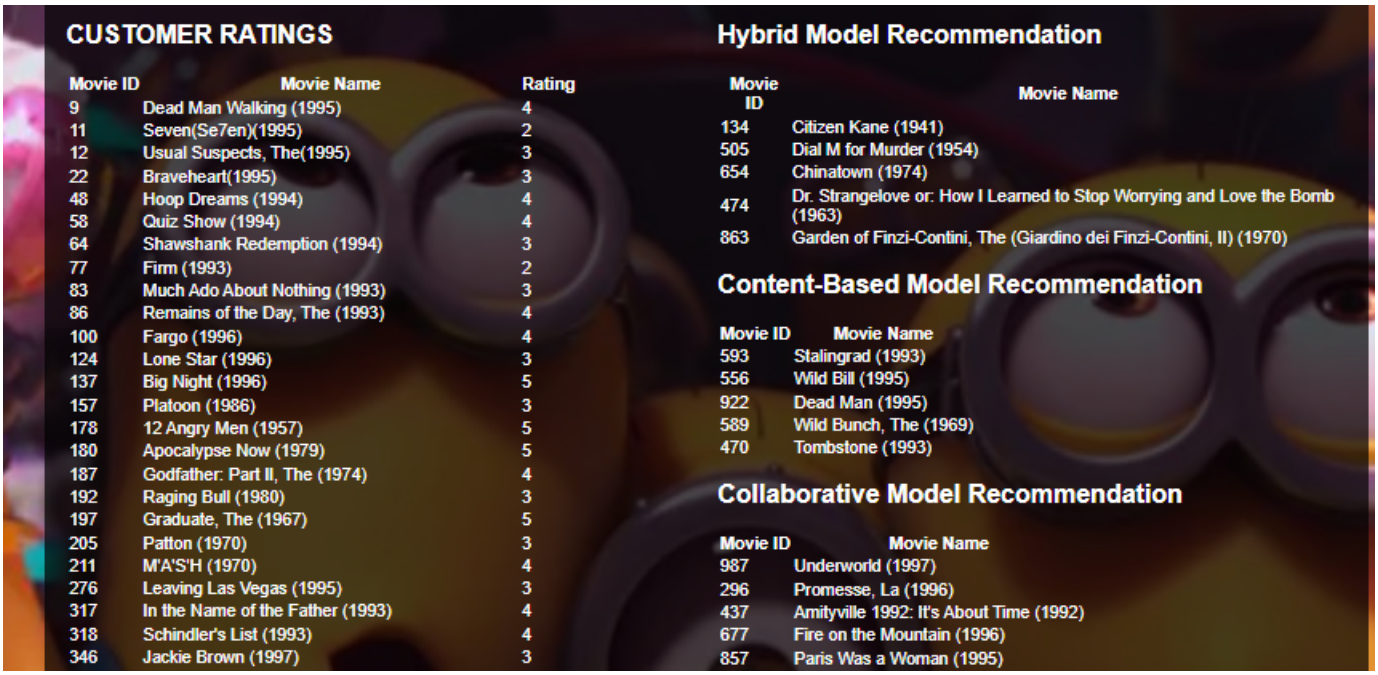


Fig. 6. Login Page UI



CUSTOMER RATINGS

Movie ID	Movie Name	Rating
9	Dead Man Walking (1995)	4
11	Seven(Se7en)(1995)	2
12	Usual Suspects, The(1995)	3
22	Braveheart(1995)	3
48	Hoop Dreams (1994)	4
58	Quiz Show (1994)	4
64	Shawshank Redemption (1994)	3
77	Firm (1993)	2
83	Much Ado About Nothing (1993)	3
86	Remains of the Day, The (1993)	4
100	Fargo (1996)	4
124	Lone Star (1996)	3
137	Big Night (1996)	5
157	Platoon (1986)	3
178	12 Angry Men (1957)	5
180	Apocalypse Now (1979)	5
187	Godfather: Part II, The (1974)	4
192	Raging Bull (1980)	3
197	Graduate, The (1967)	5
205	Patton (1970)	3
211	M'A'S'H (1970)	4
276	Leaving Las Vegas (1995)	3
317	In the Name of the Father (1993)	4
318	Schindler's List (1993)	4
346	Jackie Brown (1997)	3

Hybrid Model Recommendation

Movie ID	Movie Name
134	Citizen Kane (1941)
505	Dial M for Murder (1954)
654	Chinatown (1974)
474	Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb (1963)
863	Garden of Finzi-Contini, The (Giardino dei Finzi-Contini, Il) (1970)

Content-Based Model Recommendation

Movie ID	Movie Name
593	Stalingrad (1993)
556	Wild Bill (1995)
922	Dead Man (1995)
589	Wild Bunch, The (1969)
470	Tombstone (1993)

Collaborative Model Recommendation

Movie ID	Movie Name
987	Underworld (1997)
296	Promesse, La (1996)
437	Amityville 1992: It's About Time (1992)
677	Fire on the Mountain (1996)
857	Paris Was a Woman (1995)

Fig. 7. Recommendations

REFERENCES

[1] A. Jain and S. K. Vishwakarma. Collaborative filtering for movie recommendation using rapid miner. *International Journal of Computer Applications (0975 - 8887)*, 169(6), July 2017.

[2] S. Zhang, L. Yao, A. Sun, and Y. Tay. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)*, 52(1):1–38, 2019.

[3] Z. D Zhao and M. S Shang. User-based collaborative filtering recommendation algorithms on hadoop. In *Proc. of Third International Workshop on Knowledge Discovery and Data Mining*, pages 478–481, 2016.