

Project Report

Movie Recommender System

TMDB Dataset



**MAHARISHI DAYANAND UNIVERSITY(MDU),
ROHTAK-124001, HARYANA**

Submitted To:

Submitted by:

Urvi Sharma

B.TECH. – CSE 6th Sem

Reg. No:191390067

CERTIFICATE

This is to certify that the minor project entitled “**Movie Recommender System using TMDB Dataset** ” submitted by Urvi Sharma to Maharishi Dayanand University (MDU), Rohtak-124001, Haryana in partial fulfilment of the requirement for the award of semester 6th of the degree of B. Tech in Computer Science Engineering. This project work carried out by him under my guidance. The project fulfils the requirement as per the regulation of the institute of university and in my opinion meets the necessary standards for submission. The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university to the best of my knowledge and belief.

Date:

Mentor Signature

DECLARATION

This is to certify that the work being presented in the minor project entitled **Movie Recommender System using tmdb dataset** submitted by undersigned student of Bachelor of Technology in Computer Science Engineering to Maharishi Dayanand University (MDU), Rohtak-124001, Haryana in the fulfilment for award of semester 6th is a record of my own work carried out by me under guidance and supervision of concerned faculty and staff of the college. I certify the content of this work has not submitted elsewhere for award of any other degree.

ACKNOWLEDGMENT

I am thankful to my faculty and coordinators, for their most valuable and significant guidance throughout the course of the project and in elaborating our view of studying the project's details and getting the right vision for its implementation. Also, I would like to thank the Department of Computer Science Engineering, that has played an important role in strengthening my career by providing full cooperation and encouragement throughout the tenure of the project.

Contents

Certificate	ii
Declaration	iii
Abstract	iv
Acknowledgement	v
Table of contents	vi
List of Figures	viii
List of Tables	ix
List of Abbreviations	x
1 INTRODUCTION	1
1.1 Relevance of the Project	1
1.2 Problem Statement	2
1.3 Objective	2
1.4 Scope of the Project	3
1.5 Methodology	
2 LITERATURE SURVEY	4
2.1 k-means and k-nearest	4
2.2 Using Collaborative	5
3 SYSTEM REQUIREMENTS SPECIFICATION	6
3.1 Hardware Requirements	6
3.2 Software Specification	6
3.3 Software Requirements	6
3.3.1 Anaconda distribution	6
3.3.2 Python Libraries	7
4 SYSTEM ANALYSIS AND DESIGN	8
4.1 System Architecture	8
4.2 Activity diagram	9
4.3 Flowchart	10

5	IMPLEMENTATION	11
	5.1 Cosine similarity	11
	5.2 Singular Value Decomposition	11
	5.3 Experimental Setup	12
	Front-End/Back End implementation details	13
6	RESULTS AND DISCUSSION	14
	6.1 Screenshots	15
7	TESTING	16
	7.1 Testing Methodologies	17
8	CONCLUSION AND FUTURE SCOPE	18
	8.1 Conclusion	18
	8.2 Future Scope	18
	REFERENCES	19

Introduction

1.1 Relevance of the Project

A recommendation system or recommendation engine is a model used for information filtering where it tries to predict the preferences of a user and provide suggests based on these preferences. These systems have become increasingly popular nowadays and are widely used today in areas such as movies, music, books, videos, clothing, restaurants, food, places and other utilities. These systems collect information about a user's preferences and behaviour, and then use this information to improve their suggestions in the future.

Movies are a part and parcel of life. There are different types of movies like some for entertainment, some for educational purposes, some are animated movies for children, and some are horror movies or action films. Movies can be easily differentiated through their genres like comedy, thriller, animation, action etc. Other way to distinguish among movies can be either by releasing year, language, director etc. Watching movies online, there are a number of movies to search in our most liked movies . Movie Recommendation Systems helps us to search our preferred movies among all of these different types of movies and hence reduce the trouble of spending a lot of time searching our favourable movies. So, it requires that the movie recommendation system should be very reliable and should provide us with the recommendation of movies which are exactly same or most matched with our preferences.

A large number of companies are making use of recommendation systems to increase user interaction and enrich a user's shopping experience. Recommendation systems have several benefits, the most important being customer satisfaction and revenue. Movie Recommendation system is very powerful and important system. But, due to the problems associated with pure collaborative approach, movie recommendation systems also suffers with poor recommendation quality and scalability issues.

1.2 Problem Statement:

The goal of the project is to recommend a movie to the user.

Providing related content out of relevant and irrelevant collection of items to users of online service providers.

1.3 Objective of the Projects

- **Improving the Accuracy of the recommendation system**
- Improve the Quality of the movie Recommendation system
- **Improving the Scalability.**
- Enhancing the user experience.

1.4 Scope of the Project

The objective of this project is to provide accurate movie recommendations to users. The goal of the project is to improve the quality of movie recommendation system, such as accuracy, quality and scalability of system than the pure approaches. This is done using Hybrid approach by combining content based filtering and collaborative filtering, To eradicate the overload of the data, recommendation system is used as information filtering tool in social networking sites .Hence, there is a huge scope of exploration in this field for improving scalability,

accuracy and quality of movie recommendation systems. Movie Recommendation system is very powerful and important system. But, due to the problems associated with pure collaborative approach, movie recommendation systems also suffers with poor recommendation quality and scalability issues.

1.5 Methodology for Movie Recommendation

The hybrid approach proposed an integrative method by merging fuzzy k- means clustering method and genetic algorithm based weighted similarity measure to construct a movie recommendation system. The proposed movie recommendation system gives finer similarity metrics and quality than the existing Movie recommendation system but the computation time which is taken by the proposed recommendation system is more than the existing recommendation system. This problem can be fixed by taking the clustered data points as an input dataset

The proposed approach is for improving the scalability and quality of the movie recommendation system .We use a Hybrid approach , by unifying Content-Based Filtering and Collaborative Filtering, so that the approaches can be profited from each other. For computing similarity between the different movies in the given dataset efficiently and in least time and to reduce computation time of the movie recommender engine we used cosine similarity measure.

Agile Methodology:

1.collecting the data sets: Collecting all the required data set from Kaggle web site.in this project we require movie.csv,ratings.csv,users.csv.

2.Data Analysis: make sure that that the collected data sets are correct and analysing the data in the csv files. i.e. checking whether all the column Felds are present in the data sets.

3.Algorithms: in our project we have only two algorithms one is cosine similarity and other is single valued decomposition are used to build the machine learning recommendation model.

4.Training and Testing the model: once the implementation of algorithm is completed . we have to train the model to get the result. We have tested it several times the model is recommend different set of movies to different users.

5.Improvements in the project: In the later stage we can implement different algorithms and methods for better recommendation.

LITERATURE SURVEY

Over the years, many recommendation systems have been developed using either collaborative, content based or hybrid filtering methods. These systems have been implemented using various big data and machine learning algorithms.

2.1 Movie Recommendation System by K-Means Clustering AND K-Nearest Neighbour

A recommendation system collect data about the user's preferences either implicitly or explicitly on different items like movies. An implicit acquisition in the development of movie recommendation system uses the user's behaviour while watching the movies. On the other hand, a explicit acquisition in the development of movie recommendation system uses the user's previous ratings or history. The other supporting technique that are used in the development of recommendation system is clustering. Clustering is a process to group a set of objects in such a way that objects in the same clusters are more similar to each other than to those in other clusters. K- Means Clustering along with K-Nearest Neighbour is implemented on the movie lens dataset in order to obtain the best-optimized result. In existing technique, the data is scattered which results in a high number of clusters while in the proposed technique data is gathered and results in a low number of clusters. The process of recommendation of a movie is optimized in the proposed scheme. The proposed recommender system predicts the user's preference of a movie on the basis of different parameters. The recommender system works on the concept that people are having common preference or choice. These users will influence on each other's opinions. This process optimizes the process and having lower RMSE.

2.2 Movie Recommendation System Using Collaborative Filtering: By Ching-Seh (Mike) Wu, Deepti Garg, Unnathi Bhandary

Collaborative filtering systems analyse the user's behaviour and preferences and predict what they would like based on similarity with other users. There are two kinds of collaborative filtering systems; user-based recommender and item-based recommender.

1. **Use-based filtering:** User-based preferences are very common in the field of designing personalized systems. This approach is based on the user's likings. The process starts with users giving ratings (1-5) to some movies. These ratings can be implicit or explicit. Explicit ratings are when the user explicitly rates the item on some scale or indicates a thumbs-up/thumbs-down to the item. Often explicit ratings are hard to gather as not every user is much interested in providing feedbacks. In these scenarios, we gather implicit ratings based on their behaviour. For instance, if a user buys a product more than once, it indicates a positive preference. In context to movie systems, we can imply that if a user watches the entire movie, he/she has some likeability to it. Note that there are no clear rules in determining implicit ratings. Next, for each user, we first find some defined number of nearest neighbours. We calculate correlation between users' ratings using Pearson Correlation algorithm. The assumption that if two users' ratings are highly correlated, then these two users must enjoy similar items and products is used to recommend items to users.
2. **Item-based filtering:** Unlike the user-based filtering method, item-based focuses on the similarity between the item's users like instead of the users themselves. The most similar items are computed ahead of time. Then for recommendation, the items that are most similar to the target item are recommended to the user.

SYSTEM REQUIREMENTS SPECIFICATION

This chapter involves both the hardware and software requirements needed for the project and detailed explanation of the specifications.

3.1 Hardware Requirements

- A PC with Windows/Linux OS
- Processor with 1.7-2.4GHz speed
- Minimum of 8gb RAM
- 2gb Graphic card

3.2 Software Specification

- **Text Editor (VS-code/WebStorm)**
- Anaconda distribution package (PyCharm Editor)
- Python libraries

3.3 Software Requirements

3.3.1 Anaconda distribution:

Anaconda is a free and open-source distribution of the Python programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management system and deployment. Package versions are managed by the package management system conda. The anaconda distribution includes data-science packages suitable for Windows, Linux and MacOS.³

3.3.3 Python libraries:

For the computation and analysis we need certain python libraries which are used to perform analytics. Packages such as SKlearn, Numpy, pandas, Matplotlib, Flask framework, etc are needed.

SKlearn: It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

NumPy: NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. **Pandas:** Pandas is one of the most widely used python libraries in datascience. It provides high-performance, easy to use structures and data analysis tools. Unlike NumPy library which provides objects for multi-dimensional arrays, Pandas provides in-memory 2d table object called Data frame.

Flask: It is a lightweight WSGI web application framework. It is designed to make getting started quick and easy, with the ability to scale up to complex applications. It began as a simple wrapper around Werkzeug

SYSTEM ANALYSIS AND DESIGN

4.1 System Architecture of Proposed System:

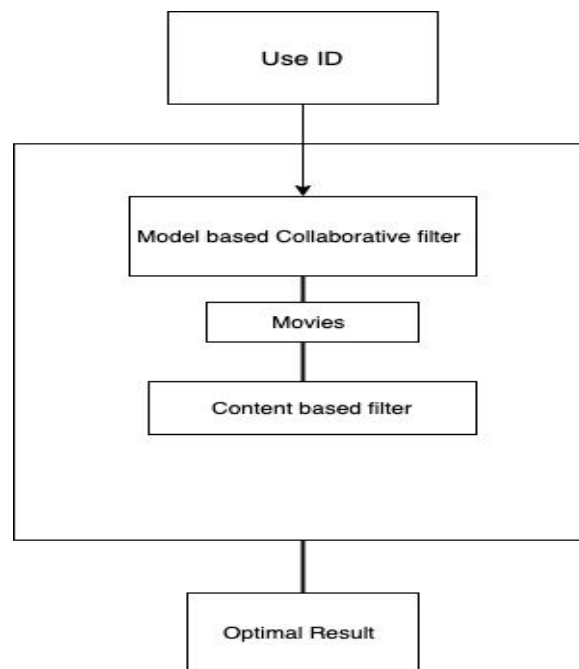


Fig:-4.1 Architecture for hybrid approach

For each different individual use different list of movies are recommended, as user login or enters the user id based on two different approaches used in the project each will recommend the set of movies to the particular user by combining the both the set of movie based on the user the hybrid model will recommend the single list of movie to the user.

Activity Diagram:

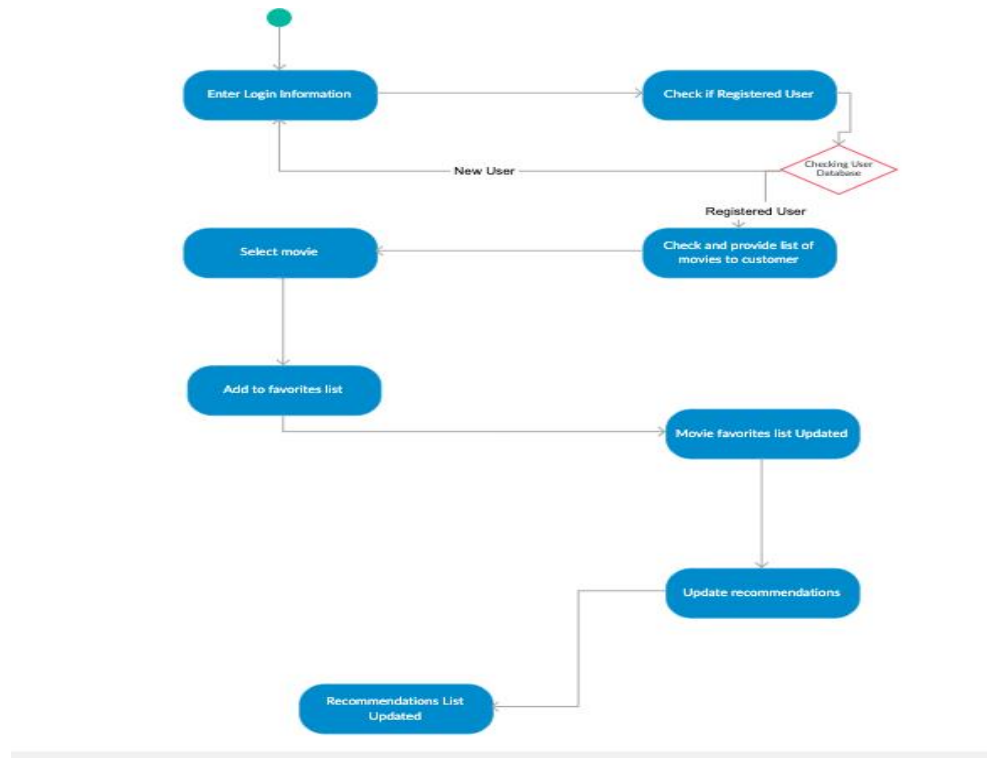


Fig:-4.2 Activity diagram

Once the user login by entering the userid i.e present in the csv file ranges from 1-5000 the list of movie are recommended to the user

4.3 Dataflow:

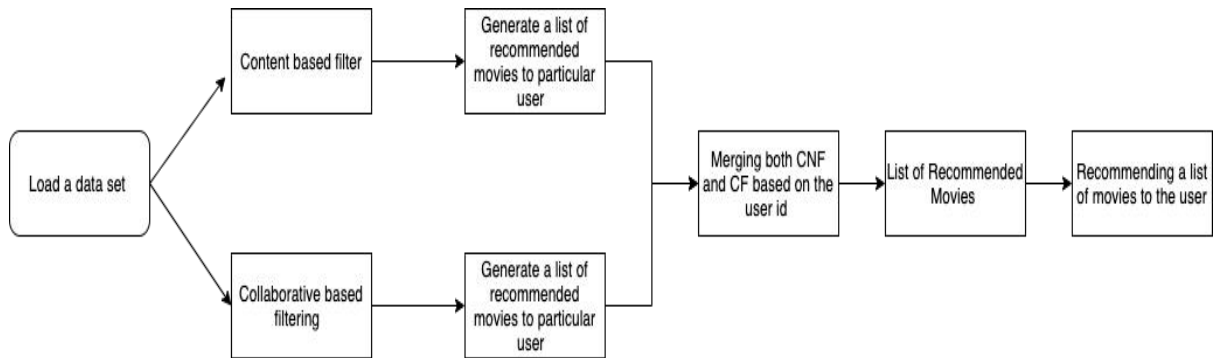


Fig:-4.3 Data Flow Diagram

Initially load the data sets that are required to build a model the data set that are required in this project are movies.csv, ratinf.csv, users.csv all the data sets are available in the Kaggle.com. Basically, two models are built in this project content based and collaborative filtering each produce a list of movies to a particular user by combining both based on the useid a single final list of movies are recommended to the particular user

Experimental requirements:

Code: Front-end (React.js)

In this project we have used popular front-end web framework (react.js) to build an interactive user interface

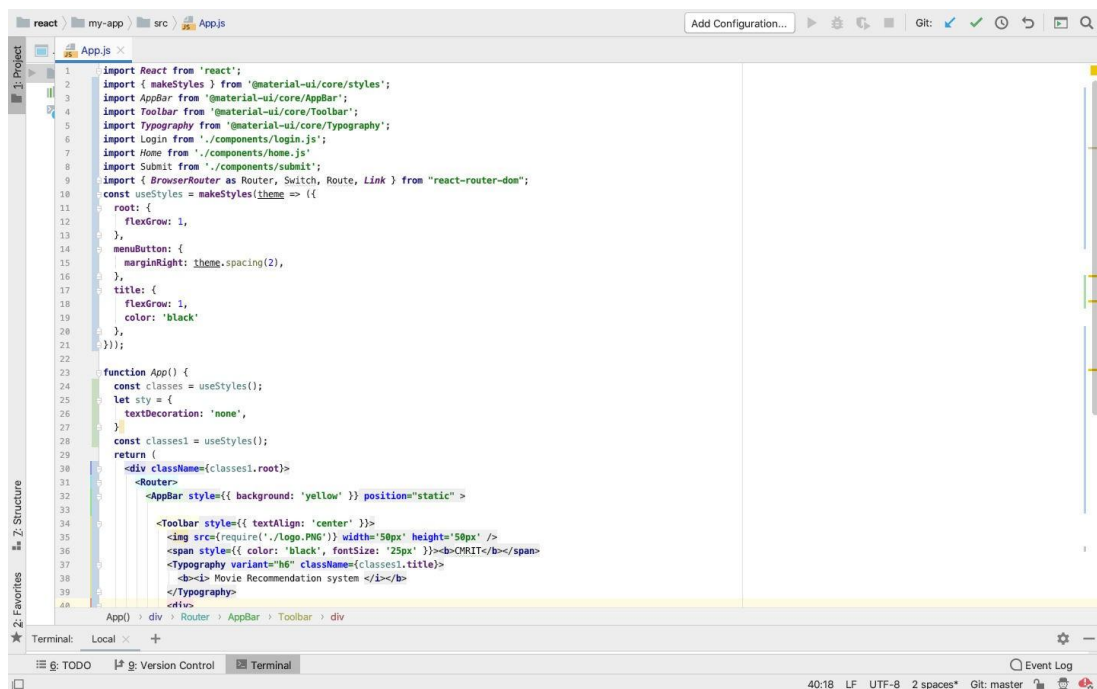
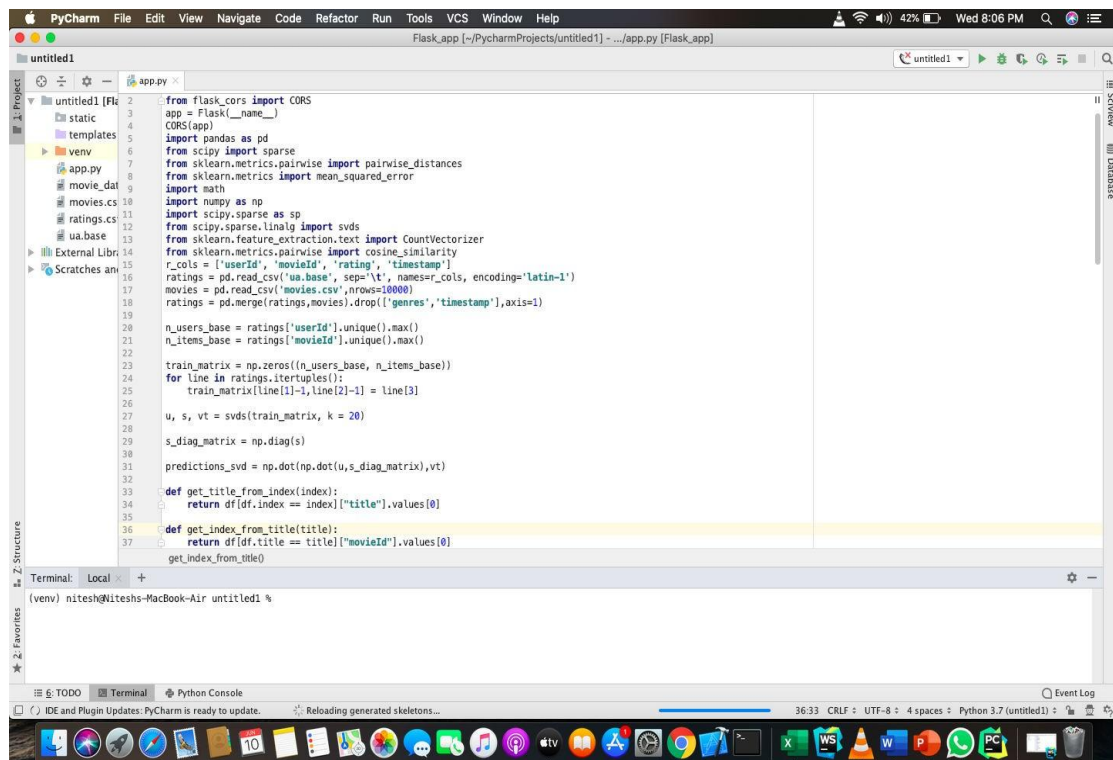


Fig:-5.1 Sample Code snippet

In react.js we used axios npm module to fetch the data from the api that is generated from flask

Backend : For backend we have use flask app to generate a local host api the resultant api is fetch in front to display the result.



```

1 from flask_cors import CORS
2 app = Flask(__name__)
3 CORS(app)
4 import pandas as pd
5 from scipy import sparse
6 from sklearn.metrics.pairwise import pairwise_distances
7 from sklearn.metrics.pairwise import mean_squared_error
8 import math
9 import numpy as np
10 import scipy.sparse as sp
11 from scipy.sparse.linalg import svds
12 from sklearn.feature_extraction.text import CountVectorizer
13 from sklearn.metrics.pairwise import cosine_similarity
14 r_cols = ['userId', 'movieId', 'rating', 'timestamp']
15 ratings = pd.read_csv('ua.base', sep='\t', names=r_cols, encoding='latin-1')
16 movies = pd.read_csv('movies.csv', nrows=10000)
17 ratings = pd.merge(ratings, movies).drop(['genres', 'timestamp'], axis=1)
18
19 n_users_base = ratings['userId'].unique().max()
20 n_items_base = ratings['movieId'].unique().max()
21
22 train_matrix = np.zeros((n_users_base, n_items_base))
23 for line in ratings.iteruples():
24     train_matrix[line[1]-1, line[2]-1] = line[3]
25
26 u, s, vt = svds(train_matrix, k=20)
27
28 s_diag_matrix = np.diag(s)
29 predictions_svd = np.dot(np.dot(u, s_diag_matrix), vt)
30
31 def get_title_from_index(index):
32     return df[df.index == index]['title'].values[0]
33
34 def get_index_from_title(title):
35     return df[df.title == title]['movieId'].values[0]
36
37 get_index_from_title
  
```

Fig:-5.2 Backed code snippet

We have developed our machine learning model in python .

By using flask, we generate resulting api which stores the data in the form of json format these data is retrieved in react by using axios npm mode and then displaying the data

RESULTS AND DISCUSSION

Since our project is movie recommendation system .one can develop a movie recommendation system by using either content based or collaborative filtering or combining both.

In our project we have developed a hybrid approach i.e combination of both content and collaborative filtering .Both the approaches have advantages and dis-advantages

.in content based filtering the it based on the user ratings or user likes only such kind of movie will recommended to the user.

Advantages: it is easy to design and it takes less time to compute

Dis-advantages: the model can only make recommendations based on existing interests of the user. In other words, the model has limited ability to expand on the users' existing interests.

In Collaborative filtering the recommendation is comparison of similar users.

Advantages: No need domain knowledge because the embeddings are automatically learned. The model can help users discover new interests. In isolation, the ML system may not know the user is interested in a given item, but the model might still recommend it because similar users are interested in that item.

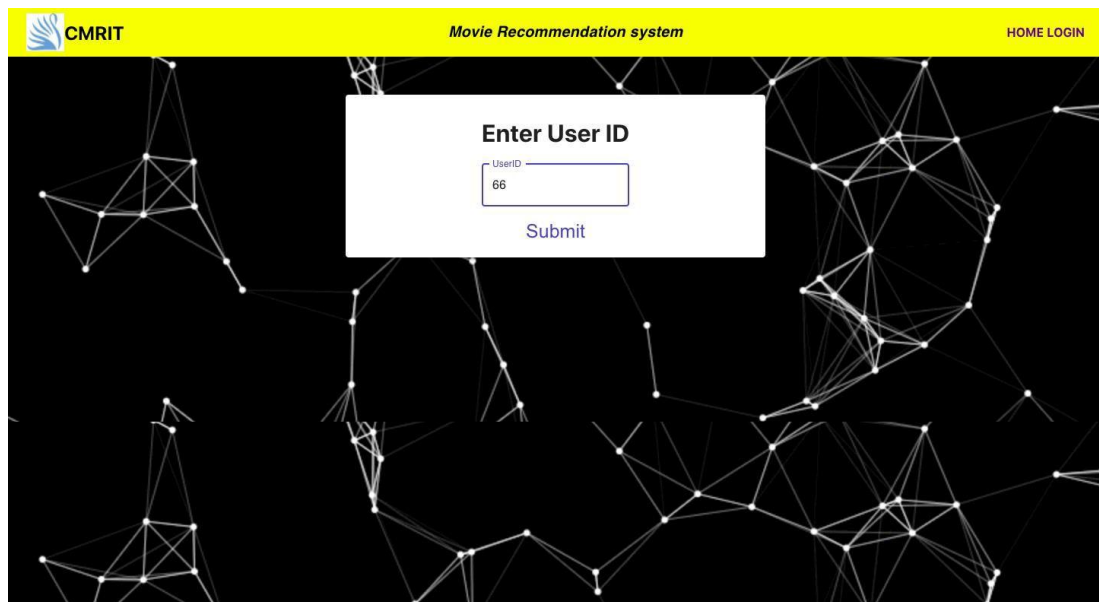
Dis-advantages: The prediction of the model for a given (user, item) pair is the dot product of the corresponding embeddings. So, if an item is not seen during training, the system can't create an embedding for it and can't query the model with this item. This issue is often called the **cold-start problem**.

The hybrid approach will resolves all these limitations by combining both content and collaborative filtering

PARAMETERS	COLLABORATIVE	CONTENT BASED	PROPOSED
	APPROACH	APPROACH	APPROACH
Accuracy	Low	Average	High
Quality	Low	Average	High
Scalability	Less	Average	High
Computing Time	Average	High	Low
Memory	Average	Low	High

The main disadvantage in hybrid approach is it require high memory

Screen shot of the result:



CMRIT Movie Recommendation system HOME LOGIN

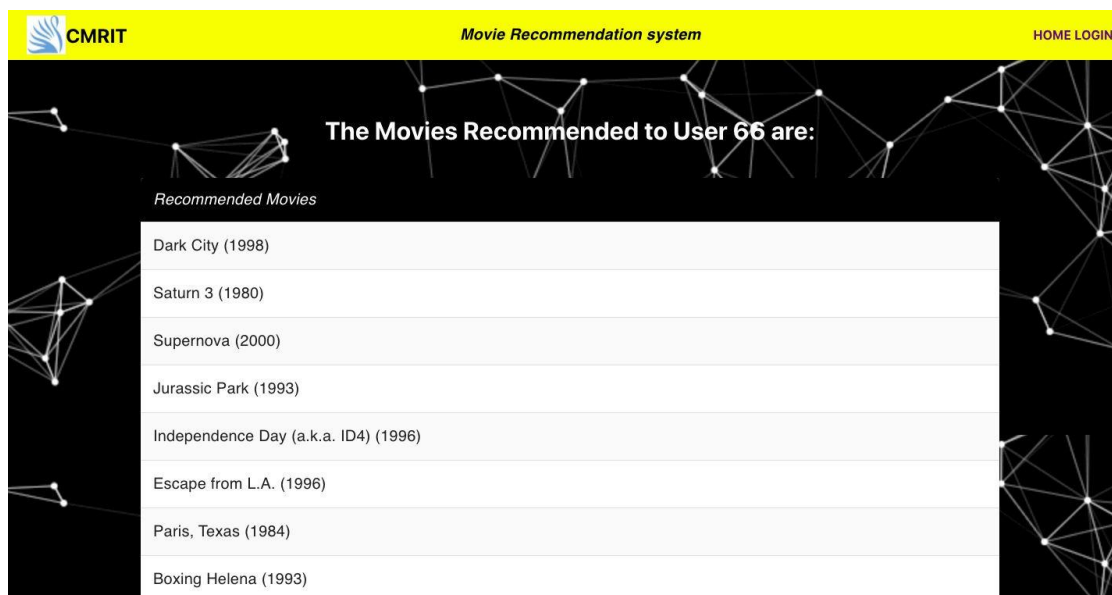
Enter User ID

UserID
66

Submit

Fig:-6.2 user id window

Enter the user id ranges between 1-5000



CMRIT Movie Recommendation system HOME LOGIN

The Movies Recommended to User 66 are:

Recommended Movies

Dark City (1998)
Saturn 3 (1980)
Supernova (2000)
Jurassic Park (1993)
Independence Day (a.k.a. ID4) (1996)
Escape from L.A. (1996)
Paris, Texas (1984)
Boxing Helena (1993)

Fig:-6.3 Display of list of recommended movies

Once the user id is entered the list of recommended movies are displayed

CONCLUSION AND FUTRURE SCOPE

8.1 Conclusion

In this project, to improve the accuracy, quality and scalability of movie recommendation system, a Hybrid approach by unifying content based filtering and collaborative filtering; using Singular Value Decomposition (SVD) as a classifier and Cosine Similarity is presented in the proposed methodology. Existing pure approaches and proposed hybrid approach is implemented on three different Movie datasets and the results are compared among them. Comparative results depicts that the proposed approach shows an improvement in the accuracy, quality and scalability of the movie recommendation system than the pure approaches. Also, computing time of the proposed approach is lesser than the other two pure approaches.

8.2 Future scope:

In the proposed approach, It has considered Genres of movies but, in future we can also consider age of user as according to the age movie preferences also changes, like for example, during our childhood we like animated movies more as compared to other movies. There is a need to work on the memory requirements of the proposed approach in the future. The proposed approach has been implemented here on different movie datasets only. It can also be implemented on the Film Affinity and Netflix datasets and the performance can be computed in the future.

REFERENCES

- [1] Hirdesh Shivhare, Anshul Gupta and Shalki Sharma (2015), “Recommender system using fuzzy c-means clustering and genetic algorithm based weighted similarity measure”, IEEE International Conference on Computer, Communication and Control.

- [2] Manoj Kumar, D.K. Yadav, Ankur Singh and Vijay Kr. Gupta (2015), “A Movie Recommender System: MOVREC”, International Journal of Computer Applications (0975 – 8887) Volume 124 – No.3.

- [3] RyuRi Kim, Ye Jeong Kwak, HyeonJeong Mo, Mucheol Kim, Seungmin Rho, Ka Lok Man, Woon Kian Chong (2015), “Trustworthy Movie Recommender System with Correct Assessment and Emotion Evaluation”, Proceedings of the International MultiConference of Engineers and Computer Scientists Vol II.

- [4] Zan Wang, Xue Yu*, Nan Feng, Zhenhua Wang (2014), “An Improved Collaborative Movie Recommendation System using Computational Intelligence”, Journal of Visual Languages & Computing, Volume 25, Issue 6.

- [5] Debadrita Roy, Arnab Kundu, (2013), “Design of Movie Recommendation System by Means of Collaborative Filtering”, International Journal of Emerging Technology and Advanced Engineering, Volume 3, Issue 4.

