IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

An Intelligent Movie Recommendation System based on user priorities

[1]Guguloth Ravinder, [2]B. Venkataramana,[3]K. Helini

[1,2,3] Assistant Professor

[1,2,3] Department of Computer Science and Engineering,
[1,2] Holy Mary Institute of Technology & Science, Bogaram, Hyderabad, Telangana, India.
[3]Vignan's Institute of Management and Technology for Women, Bogaram, Hyderabad, Telangana, India.

Abstract: Today's web and app users request modified experiences. They anticipate the apps, news sites, social networks they engage with to evoke who they are and what they're fascinated in and make related, adjusted, and accurate commendations for new content and new goods based on their earlier deeds. This can be done using Recommended Systems in Machine Learning. In this paper we use Recommender System to recommend movies based on his previous ratings on movie he came across.

***Index Terms* - Recommended System, Machine Learning, Rating, Movies, One hot encoding.**

I. Introduction

In the current period of advancing web technologies, the recommender systems (RSs) have revolved the notice of the commercial society and the common man towards itself due to its consequence and importance in the e-commerce. Nowadays e-commerce is believed to be intensely connected to the customer's satisfaction, and an vital achievement is always dependent on customer dependability. The same is with the online booking systems being a core component of the tourism industry. The main purpose of our recommender system is to provide suggestions and recommendations which are truly based on customer's preference and choice. Recommendation system (RS) helps customers not only finding appropriate movies but also it is benefitting in all domains such as hostels, books, and all sorts of other different products and items. Different types of data i.e., hotels, movies, and music, can be processed by RS.

The main idea here is to develop a recommender system which helps users to find movies according to their preference and choice using previous users' reviews and ratings. When we are dealing with the recommending a movie to the user we mainly focus on the rating given by the user for the movies he/she was interested in. Based on the rating we select which type of genre is mostly liked by the user. Using this user's genre choice we will recommend movies which contain genre the user is interested.

The rest of the paper is divided into four sections. Section 2 introduces types of Recommended System. The Section 3 is for Preprocessing of data, in which we identify the categorical variables. This section also identifies the variables that are not relevant or not contributing more to the analysis. The Section 4 undergoes the process of recommendation system on movie data. . Section V is a conclusion on the results presented in previous section.

II. TYPES OF RECOMMENDATION SYSTEM

A Recommendation System is a software tool designed to make and deliver suggestions for things or content a user would like to purchase. Using machine learning techniques and various data about individual products and individual users, the system creates an advanced net of complex connections between those products and those people. These are a collection of algorithms used to recommend items to users based on information taken from the user. These systems have become ubiquitous, and can be commonly seen in online stores, movies databases and job finders. There are 3 types of recommendation systems

1. Popularity based recommendation engine
2. Content based recommendation engine
3. Collaborative filtering based recommendation engine

Popularity based recommendation engine:

Popularity based is the simplest kind of recommendation engine that you will come across. The trending list you see in YouTube or Netflix is based on this algorithm. It keeps a track of view counts for each movie/video and then lists movies based on views in descending order.

Content based recommendation engine:

Content based recommendation engine takes in a movie that a user currently likes as input. Then it analyzes the contents of the movie to find out other movies which have similar content. Then it ranks similar movies according to their similarity scores and recommends the most relevant movies to the user.

Collaborative filtering based recommendation engine:

Collaborative filtering based recommendation engine first tries to find similar users based on their activities and preferences. Now, between these users (say, A and B) if user A has seen a movie that user B has not seen yet, then that movie gets recommended to user B and vice-versa. In other words, the recommend-dations get filtered based on the collaboration between similar user's preferences. One typical application of this algorithm can be seen in the Amazon e-commerce platform, where you get to see the "Customers who viewed this item also viewed" and "Customers who bought this item also bought" list.

III. METHODOLOGY

In this section, we will invoke the data, preprocess the data and visualize the data.

3.1 Data and its features

The datasets used in this system is taken from "GroupLens" namely "movie.csv" and "rating.csv". The movie.csv contains features like movieid, movie name and its genre. The rating dataset contains features like userid, movieid and rating given by the user to that movie.
First thing to do is to import necessary libraries as shown in Fig 1. Then we should invoke data from the datasets using pandas as shown in Fig 2. The features of the datasets are presented in Fig 3 and Fig 4.

```
#Dataframe manipulation library
import pandas as pd
import numpy as np
#Math functions
from math import sqrt
#Visualizing
import plotly.express as px
```

Fig 1: Importing libraries

```
#importing datasets
movies = pd.read_csv("E:/Project's Datasets/Movie Recommendation System/movies.csv")
ratings = pd.read_csv("E:/Project's Datasets/Movie Recommendation System/ratings.csv")
```

Fig 2: Invoking data

```
#Features of movie dataset
movies.head()
```

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II (1995)	Comedy

Fig 3: Features of movie dataset

```
#features of rating dataset
ratings.head()
```

	userId	movieId	rating	timestamp
0	1	296	5.0	1147880044
1	1	306	3.5	1147868817
2	1	307	5.0	1147868828
3	1	665	5.0	1147878820
4	1	899	3.5	1147868510

Fig 4: Features of rating dataset

3.2 Data Prepossessing

Data preparation is the first step in data analytics projects and can include many discrete tasks such as loading data or data ingestion, data fusion, data cleaning, data augmentation, and data delivery. In movie data we can remove year from the title feature and add a column of it to the dataset. This is shown in Fig 5.

```
#Cleaning of movie dataset for analyzing
#Seperting year from title column
movies['year'] = movies.title.str.extract('(\(\d\d\d\d\))',expand=False)
#Removing the parentheses
movies['year'] = movies.year.str.extract('(\d\d\d\d)',expand=False)
#Removing the years from the 'title' column
movies['title'] = movies.title.str.replace('(\(\d\d\d\d\))', '')
#Applying the strip function to get rid of any ending whitespace characters that may have appeared
movies['title'] = movies['title'].apply(lambda x: x.strip())
```

```
#modified movie dataset
movies.head()
```

	movieId	title	genres	year
0	1	Toy Story	Adventure|Animation|Children|Comedy|Fantasy	1995
1	2	Jumanji	Adventure|Children|Fantasy	1995
2	3	Grumpier Old Men	Comedy|Romance	1995
3	4	Waiting to Exhale	Comedy|Drama|Romance	1995
4	5	Father of the Bride Part II	Comedy	1995

Fig 5: Preprocessing of movie dataset

Genre feature is a categorical variable so we should convert the list of genres to a vector where each column corresponds to one possible value of the feature. This can be done by using One Hot Encoding technique. This encoding is needed for feeding categorical data. In this case, we store every different genre in columns that contain either 1 or 0. 1 shows that a movie has that genre and 0 shows that it doesn't. This is shown in Fig 6.

While coming to rating dataset the feature 'timestamp' is not useful for recommending a movie so we can drop it from the dataset as shown in Fig 7.

	movieId	title	genres	year	Adventure	Animation	Children	Comedy	Fantasy	Romance	...	Horror	Mystery	Sci-Fi	IMAX
0	1	Toy Story	[Adventure, Animation, Children, Comedy, Fantasy]	1995	1.0	1.0	1.0	1.0	1.0	0.0	...	0.0	0.0	0.0	0.0
1	2	Jumanji	[Adventure, Children, Fantasy]	1995	1.0	0.0	1.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0
2	3	Grumpier Old Men	[Comedy, Romance]	1995	0.0	0.0	0.0	1.0	0.0	1.0	...	0.0	0.0	0.0	0.0
3	4	Waiting to Exhale	[Comedy, Drama, Romance]	1995	0.0	0.0	0.0	1.0	0.0	1.0	...	0.0	0.0	0.0	0.0
4	5	Father of the Bride Part II	[Comedy]	1995	0.0	0.0	0.0	1.0	0.0	0.0	...	0.0	0.0	0.0	0.0

Fig 6: Data frame after using One Hot Encoding

```
#Cleaning rating Dataset by dropping unnecessary features
ratings = ratings.drop('timestamp', 1)
ratings.head()
```

	userId	movieId	rating
0	1	296	5.0
1	1	306	3.5
2	1	307	5.0
3	1	665	5.0
4	1	899	3.5

Fig 7: Final rating dataset.

3.3 Data Visualization

Data visualization is the graphical representation of information and data. By using visual elements like charts, graphs, and maps, data visualization tools provide an accessible way to see and understand trends, outliers, and patterns in data. The statistical analysis of no .of movies are rated for a rate can be show by using histogram as shown in Fig 8.

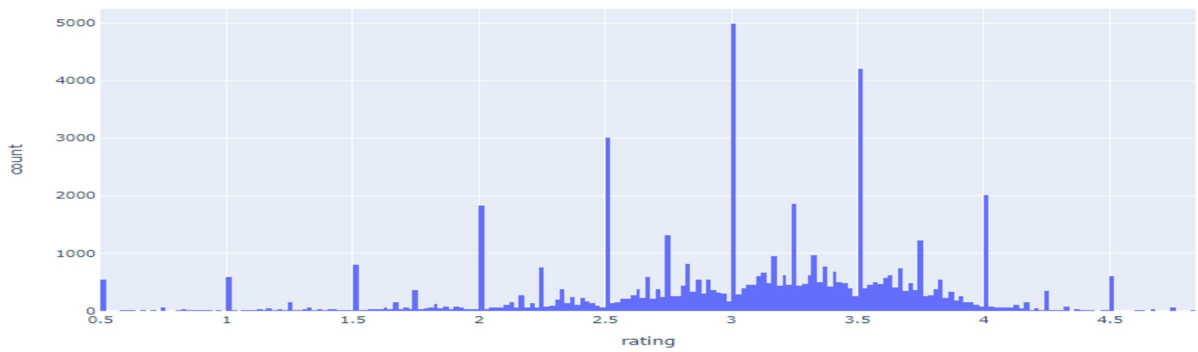

Fig 8: Statistical analysis of rating.

IV. CONTENT-BASED RECOMMENDATION SYSTEM

Now, let's take a look at how to implement Content-Based or Item-Item recommendation systems. This technique attempts to figure out what a user's favorite aspects of an item is, and then recommends items that present those aspects. In our case, we're going to try to figure out the input's favorite genres from the movies and ratings given. Let's begin by creating an input user to recommend movies and is shown in the following Fig 9. After adding movie id to the user preferred movies the data set is as shown in Fig 10.

```
#user rated movies
userInput = [
            {'title':'Star Kid', 'rating':4.4},
            {'title':"Monster's Ball", 'rating':3.7},
            {'title':'Big Heat, The', 'rating':4.0},
            {'title':"Bossa Nova", 'rating':5},
            {'title':'Maximum Overdrive', 'rating':4.5}
         ]
inputMovies = pd.DataFrame(userInput)
inputMovies = inputMovies[['title','rating']]
inputMovies
```

	title	rating
0	Star Kid	4.4
1	Monster's Ball	3.7
2	Big Heat, The	4.0
3	Bossa Nova	5.0
4	Maximum Overdrive	4.5

Fig 9: User's data

	movieId	title	rating
0	1750	Star Kid	4.4
1	2119	Maximum Overdrive	4.5
2	3567	Bossa Nova	5.0
3	5015	Monster's Ball	3.7
4	5017	Big Heat, The	4.0

Fig 10: User's data with movieid

We're going to start by learning the input's preferences, so let's get the subset of movies that the input has watched from the Data frame containing genres defined with binary values as shown in Fig 11.

We'll only need the actual genre table, so let's clean this up a bit by resetting the index and dropping the movieId, title, genres and year columns which is explained in fig12.

```python
#User movie's genre
userMovies = moviesWithGenres_df[moviesWithGenres_df['movieId'].isin(inputMovies['movieId'].tolist())]
userMovies
```

	movieId	title	genres	year	Adventure	Animation	Children	Comedy	Fantasy	Romance	...	Horror	Mystery	Sci-Fi	IMAX	Documentary
1680	1750	Star Kid	[Adventure, Children, Fantasy, Sci-Fi]	1997	1.0	0.0	1.0	0.0	1.0	0.0	...	0.0	0.0	1.0	0.0	0.0
2029	2119	Maximum Overdrive	[Horror]	1986	0.0	0.0	0.0	0.0	0.0	0.0	...	1.0	0.0	0.0	0.0	0.0
3468	3567	Bossa Nova	[Comedy, Drama, Romance]	2000	0.0	0.0	0.0	1.0	0.0	1.0	...	0.0	0.0	0.0	0.0	0.0
4909	5015	Monster's Ball	[Drama, Romance]	2001	0.0	0.0	0.0	0.0	0.0	1.0	...	0.0	0.0	0.0	0.0	0.0
4911	5017	Big Heat, The	[Drama, Film-Noir]	1953	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0

5 rows × 24 columns

Fig 11: User's movie with its genre

```python
#Resetting the index to avoid future issues
userMovies = userMovies.reset_index(drop=True)
#Dropping unnecessary issues due to save memory and to avoid issues
userGenreTable = userMovies.drop('movieId', 1).drop('title', 1).drop('genres', 1).drop('year', 1)
userGenreTable
```

	Adventure	Animation	Children	Comedy	Fantasy	Romance	Drama	Action	Crime	Thriller	Horror	Mystery	Sci-Fi	IMAX	Documentary	War	Musical
0	1.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	1.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Fig 12: User Genre table

Now we should learn about the input preferences. To do this first we should convert genre into weights. We can do this by using the input's reviews and multiplying them into the input's genre table and then summing up the resulting table by column. Actually we are applying dot product between a matrix and a vector. The output to this is shown in Fig 13.

```python
#User's Profile
userProfile = userGenreTable.transpose().dot(inputMovies['rating'])
userProfile
```

```
Adventure              4.4
Animation              0.0
Children               4.4
Comedy                 5.0
Fantasy                4.4
Romance                8.7
Drama                 12.7
Action                 0.0
Crime                  0.0
Thriller               0.0
Horror                 4.5
Mystery                0.0
Sci-Fi                 4.4
IMAX                   0.0
Documentary            0.0
War                    0.0
Musical                0.0
Western                0.0
Film-Noir              4.0
(no genres listed)     0.0
dtype: float64
```

Fig 13: User's profile in terms of genre.

Using this, we can recommend movies that satisfy the user's preferences. Now let's get the genre table of all the movies which is displayed in Fig 14.

```python
#Genre table from original table
genreTable = moviesWithGenres_df.set_index(moviesWithGenres_df['movieId'])
#Drop the unnecessary features
genreTable = genreTable.drop('movieId', 1).drop('title', 1).drop('genres', 1).drop('year', 1)
genreTable.head()
```

movieId	Adventure	Animation	Children	Comedy	Fantasy	Romance	Drama	Action	Crime	Thriller	Horror	Mystery	Sci-Fi	IMAX	Documentary	War
1	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	1.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	1.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Fig 14: Genre table of all the movies

With the input's profile and the complete list of movies and their genres in hand, we're going to take the weighted average of every movie based on the input profile and recommend the top ten movies that most satisfy which is shown in Fig 15.

```
#Recomendation table
recommendationTable_df = ((genreTable*userProfile).sum(axis=1))/(userProfile.sum())
recommendationTable_df = recommendationTable_df.sort_values(ascending=False)
recommendationTable_df.head()
```

```
movieId
146305    0.754286
172357    0.754286
26093     0.754286
49593     0.744762
111523    0.672381
dtype: float64
```

Fig 15: Recommendation table

V. CONCLUSION

By using Recommended System we predicted top 20 movies based on the requirements of the user and the output is shown in Fig 16. This is to conclude that recommended presented in this paper is very helpful to study the priorities of the customers and recommend other movies similar to their interest.

	movieId	title	genres	year
1818	1907	Mulan	[Adventure, Animation, Children, Comedy, Drama...	1998
8571	26093	Wonderful World of the Brothers Grimm, The	[Adventure, Animation, Children, Comedy, Drama...	1962
9177	27344	Revolutionary Girl Utena: Adolescence of Utena...	[Action, Adventure, Animation, Comedy, Drama, ...	1999
10830	45672	Click	[Adventure, Comedy, Drama, Fantasy, Romance]	2006
11241	49593	She	[Action, Adventure, Drama, Fantasy, Horror, Ro...	1965
13905	71999	Aelita: The Queen of Mars (Aelita)	[Action, Adventure, Drama, Fantasy, Romance, S...	1924
21632	111523	50 Kisses	[Comedy, Drama, Horror, Romance, Sci-Fi]	2014
25437	123731	Mad About Men	[Comedy, Drama, Fantasy, Romance, Sci-Fi]	1954
25790	124519	Snow White and the Three Stooges	[Adventure, Children, Comedy, Drama, Fantasy]	1961
26399	125972	Halloweentown II: Kalabar's Revenge	[Adventure, Children, Comedy, Drama, Fantasy]	2001
29850	134853	Inside Out	[Adventure, Animation, Children, Comedy, Drama...	2015
30113	135512	Three Wishes for Cinderella	[Adventure, Children, Drama, Fantasy, Romance]	1973
34813	146305	Princes and Princesses	[Animation, Children, Comedy, Drama, Fantasy, ...	2000
35827	148775	Wizards of Waverly Place: The Movie	[Adventure, Children, Comedy, Drama, Fantasy, ...	2009
41905	162700	1st Bite	[Comedy, Drama, Fantasy, Horror, Romance]	2006
46020	171591	You're So Cupid	[Children, Drama, Fantasy, Romance, Sci-Fi]	2010
46396	172357	Deal of a Lifetime	[Children, Comedy, Drama, Fantasy, Romance, Sc...	1999
49842	179729	Prince Charming	[Adventure, Children, Comedy, Fantasy, Romance...	2001
50013	180091	Pokémon the Movie: I Choose You!	[Adventure, Animation, Children, Comedy, Drama...	2017
58935	199736	Naruto the Movie: Legend of the Stone of Gelel	[Adventure, Animation, Comedy, Drama, Fantasy,...	2005

Fig 16: Final Recommended Movies

REFERENCES

[1] M. Ibrahim and I. Bajwa, "Design and application of a multi-variant expert system using Apache Hadoop framework," Sustainability, vol. 10, no. 11, p. 4280, 2018.

[2] M.-Y. Hsieh, W.-K. Chou, and K.-C. Li, "Building a mobile movie recommendation service by user rating and APP usage with linked data on Hadoop," Multimedia Tools and Applications, vol. 76, no. 3, pp. 3383–3401, 2017.

[3] Z. Tan and L. He, "An efficient similarity measure for user-based collaborative filtering recommender systems inspired by the physical resonance principle," IEEE Access, vol. 5, pp. 27211–27228, 2017.

[4] T. Chen and Y. H. Chuang, "Fuzzy and nonlinear programming approach for optimizing the performance of ubiquitous hotel recommendation," Journal of Ambient Intelligence and Humanized Computing, vol. 9, no. 2, pp. 275–284, 2018.

[5] Y. H. Hu, P. J. Lee, K. Chen, J. M. Tarn, and D.-V. Dang, "Hotel recommendation system based on review and context information: a collaborative filtering appro," in Proceedings of the Pacific Asia Conference on Information Systems PACIS, p. 221, Chiayi City, Taiwan, June-July 2016.

[6] R. Sandeep, S. Sood, and V. Verma, "Twitter sentiment analysis of real-time customer experience feedback for predicting growth of Indian telecom companies," in Proceedings of the 2018 4th International Conference on Computing Sciences (ICCS), pp. 166–174, IEEE, Phagwara, India, August 2018.

[7] M. Ibrahim, I. S. Bajwa, R. Ul-Amin, and B. Kasi, "A neural network-inspired approach for improved and true movie recommendations," Computational Intelligence and Neuroscience, vol. 2019, no. 7, Article ID 4589060, 19 pages, 2019.