EFFECTIVE MOVIE RECOMMENDATION SYSTEMS

Project report submitted

in partial fulfilment of the requirement for the degree of

M.Tech Integrated

By

B V N SRAVYA	-	20MIC0093
R. VENKATA TAGORE REDDY	-	20MIC0046
D. NITHIN SRI SARVESWAR	-	20MIC0038
SUDA NAVEEN	-	20MIC0057

Under

Web Mining and Social Network Analysis

CSI3033

J Component

Department of Computational Intelligence

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF COMPUTER SCIENCE & ENGINEERING

VELLORE INSTITUTE OF TECHNOLOGY, VELLORE

APRIL 2023

TABLE OF CONTENTS

ABSTRACT

This project proposes Recommender systems are efficient tools for filtering online information, which is widespread owing to the changing habits of computer users, personalization trends, and emerging access to the internet. Even though the recent recommender systems are eminent in giving precise recommendations, they suffer from various limitations and challenges like scalability, cold-start, sparsity, etc. Due to the existence of various techniques, the selection of techniques becomes a complex work while building application-focused recommender systems. In addition, each technique comes with its own set of features, advantages and disadvantages which raises even more questions, which should be addressed. This project aims to undergo a systematic review on various recent contributions in the domain of recommender systems, focusing on diverse applications like books, movies, products, etc. Initially, the various applications of each recommender system are analyzed. Then, the algorithmic analysis on various recommender systems is performed and a taxonomy is framed that accounts for various components required for developing an effective recommender system. In addition, the datasets gathered, simulation platform, and performance metrics focused on each contribution are evaluated and noted. Finally, this review provides a much-needed overview of the current state of research in this field and points out the existing gaps and challenges to help posterity in developing an efficient recommender system. Specifically the development of recommendation system has been confined to only movie field in this project. Datasets used in this project are taken from the standard movie lens and IMDB websites. The idea is to make use of filtering and clustering techniques to suggest items of interest to users. For a media commodity like movies, suggestions are made to users by finding user profiles of individuals with similar tastes. Initially, user preference is obtained by letting them rate movies of their choice. Upon usage, the recommender system will be able to understand the user better and suggest movies that are more likely to be rated higher. The experiment results on the MovieLens dataset provides a reliable model which is precise and generates more personalised movie recommendations compared to other models. In implementation part hybrid approach is supported with Natural Language Processing method to get personalized recommendations. The experiment results obtained on Movielens dataset stipulate that the proposed approach may provide high performance regarding reliability, efficiency and delivers accurate personalized movie recommendations when compared with existing methods.

Keywords-Movies, Recommendation system, hybrid approach, Natural Language Processing.

.

Chapter 1: INTRODUCTION

1.1 Introduction

Recommender systems primarily aim to reduce the user's effort and time required for searching relevant information over the interne User's profile (with relevant personal information as well as search history of a user) is compared with search history of different users for a few attributes of the item user is looking for and on the basis of 'ratings' or 'preferences' given by different users for that item, recommender system predicts the recommendations for that item. The filtering techniques have been categorized as 1) Content-based, 2) Collaborative and 3) Hybrid. These are discussed below:

A. The Content-based Approach

Content-based filtering technique works with profiles of users. A profile has information about a user and his preferences. Preferences rely on how the user rated items and what item user has bought and viewed. Generally, once profile is made, recommender systems create a survey, to get basic information about a user so as to avoid the new-user problem. New-user problem arises when profile of the same user is made having different attributes. Profiles are obtained by analyzing the items previously seen and rated by the user and are typically created using keyword analysis techniques from information retrieval. Within the recommendation method, the engine compares the things that were already positively rated by the user.

B. The Collaborative Approach

In collaborative technique, the concept of neighborhood is used. Here, neighbors are the group of people who have rated same item and have similar taste. Items are recommended which have not been rated by the user but were already positively rated by users in his neighborhood that didn't rate and appears for similarities.

We can distinguish the collaborative filtering on basis of
following famous approaches: i) user-based ii) item-based.

i.User-based approach

The main role is performed by user in the user-based approach. Items will be recommended to the user based on what is viewed by other user from the same group.

Recommendations are given to the user based on evaluation of items by other users from the same community, with whom user shares common preferences. If the community rates the item positively, it will be suggested to the user. Thus in the user-based approach the items that were already rated by the user plays an important role in searching a group that shares preferences with User's profile (with relevant personal information as well as search history of a user) is compared with search history of different users for a few attributes of the item user is looking for and on the basis of 'ratings' or 'preferences' given by different users for that item, recommender system predicts the recommendations for that item. The filtering techniques have been categorized as Content-based, Collaborative and Hybrid. These are discussed below:

A. The Content-based Approach

Content-based filtering technique works with profiles of users. A profile has information about a user and his preferences. Preferences rely on how the user rated items and what item user has bought and viewed. Generally, once profile is made, recommender systems create a survey, to get basic information about a user so as to avoid the new-user problem. New-user problem arises when profile of the same user is made having different attributes. Profiles are obtained by analyzing the items previously seen and rated by the user and are typically created using keyword analysis techniques from information retrieval. Within the recommendation method, the engine compares the things that were already positively rated by the user with the things that didn't rate and appears for similarities. We can distinguish the collaborative filtering on basis of following famous approaches: i) user-based ii) item-based.

i. User-based approach

The main role is performed by user in the user-based approach. Items will be recommended to the user based on what is viewed by other user from the same group. Recommendations are given to the user based on evaluation of items by other users from the same community, with whom user shares common preferences. If the community rates the item positively, it will be suggested to the user. Thus in the user-based approach the items that were already rated by the user plays an important role in searching a group that shares preferences with him. ii. Item-based approach

It is fact that the preferences of users remain constant or change very slightly so the similar items build neighborhoods based on appreciations of users. Afterwards the system generates recommendations of items in the neighborhood that a user will prefer.

C The Hybrid Approach

This approach combines the multiple filtering methods to get a refined result. Different criteria to combine different recommendation filtering techniques are classified as:

C The Hybrid Approach

This approach combines the multiple filtering methods to get a refined result. Different criteria to combine different recommendation filtering techniques are classified as:

1. Separately implementing content-based methods and merging their predictions

2. Integrate some content-based characteristics into a collaborative approach

3. Integrate some collaborative characteristics into a content-based approach, and

4. Developing a general integrated model that merges characteristics of both content-based and collaborative filtering methods. It reduces the filtering problems as advantages of one technique can be used to minimize the disadvantage of other technique.

1.2 Motivation

The motivation of this problem is to recommend the users with the most appropriate movies and maintain the accuracy score of the recommendation system.

1.3 Brief Overview of Problem

The brief overview of Problem is to provide users with personalized recommendations and top-N recommendations entitled as recommended for you.

1.4 Scope of the Project

The project focuses on a recommendation engine that filters the data using different algorithms and recommends the most relevant items to users. It can achieve the target to recommend via text as well as speech.

1.5 Significant Contribution

Our significant contribution towards this field is to embedding the recommendations with text as well as speech.

Chapter 2: REVIEW OF LITERATURE

2.1 Reviews

Recommendation systems facilitate users by using supplying useful suggestions, consequently lowering their seek time. The consumer rankings are used for classifying information into numerous classes that could in addition be useful to generate hints. In this, we're going to use statistics mining techniques to examine consumer preferences and determine consumer-unique film ratings via the help of information mining techniques [2]. We can use a film database from IMDB and decide user particular scores for each of them. The analysis of attributes of those films will assist us to become aware of the decisive factors and discover consumer alternatives appropriately [4]. This project introduces a recommendation algorithm based on Gaussian NBand applies it to the recommendation of movies by mining user behavior data and recommending movies with higher ratings to them. This article used the data provided by the movie website Movie Lens. It is the testing set and training set that the data is divided into, and the Top-N recommendation list is produced for the training set, while the algorithm is evaluated on the testing set [5]. It is the features of the data that Gaussian NB can effectively extract and complete the recommendation from the results. According to this paper, the modern optimization for traditional recommendation systems could be summarized as follows. To strengthen user control [1].

Most existing recommendation system according to preset automatically generates multiple recommended users' personal information and requirements, to some extent, limits user participation and control [8]. The result of the recommendation system should allow users to participate in parameter definition. Recommendation system can be achieved by the relevance feedback mechanism to update the user's real-time demand, for example, by the user to explicitly to the evaluation of the recommended collect user feedback information. Recommender systems use algorithms to provide users with useful recommendations for products or services [5]. Recently, these systems have been using machine learning algorithms from the field of artificial intelligence. Recommendation system is categorized in four classes: Simple System using popularity, Collaborative Filtering, Content based and hybrid based Approach [10]. A movie recommendation is being highly recognized as a part of our social life due to its strength in providing enhanced entertainment. Such a system will predict what movies a user will like based on various criterias like the attributes of previously liked movies by that

user or the popularity of the movies. Although a set of movie recommendation systems have been proposed and implemented [9].

2.2 Research Gaps

Sparsity Problem :

In the recommendation system, it has been noticed that most users use the system but do not give rating for feedback to the system in a proper way. So even though we may have many users using the recommender system various Times, it is possible that we have very few ratings from those users about different items which they have liked or purchased or even disliked. Rating seems useless to users, so they avoid giving it and sometimes may lead to false rating such as providing 5 stars(Considered Best for them) or 1 stars (Considered non likely item), without even noticing what kind of product it is. This is taken as input by the recommender system, which further displays the unwanted results to the user, which may lose the interest of the user in that platform, and lead to non-efficient working. As rating has no significance as per the users, they sometimes don't even give to bulk of products, which again leads to the same problem. Sparsity problem is a problem which arises due to sparsity of rating matrix. here we are talking about the user-item rating matrix. In mathematical terms user-item rating matrix is sparse, It gives rise to a unique problem known as sparsity problem in recommender systems.

Scalability:

Scalability is the property of the system which defines weather system will be able to cope up when the system grows.For example, in case of recommender systems, scalability can be understood as a situation where a recommender system is performing very well in case of few users like 1000 users but as the user grows to 10000 or 100000 it starts performing a way which is not desirable. When the system faces scalability issues it becomes slow it starts feeling it start giving problems which it has never given when a load of users recommendation were less. The scalability issues can be divided into two parts hardware scalability and software scalability. the hardware is scalability is about the increase of hardware to solve the scalability problem. For example, one can increase processor, RAM and server configuration to solve the problem. But only hardware air capacity increase cannot solve the problem.

Chapter 3: PROBLEM DEFINITION

3.1 Problem Statement

The problem statement is trying to forecast the opinion the users will have on the dissimilar substance and be able to recommend the finest items to each user.

3.2 Problem Definition

The problem definition is to find the similar users to recommend the users with top-N recommendations and try to fulfil the gaps as mentioned earlier in the report. We designed customized algorithm so that it can overcome the problems of scalability and sparsity. We have taken 100000 rows for training and it has taken less than a minute to get executed and showcase the results. Sparsity is the major problem of SVD ,so we have taken collaborative filtering to eliminate the zeros from the rating matrix. For personalised recommendations , NLP method like gaussian NB is used to extract emotions from the reviews text file.

3.3 Objectives

- Movie recommendation system aims to provide users with accurate movie recommendations
- Movie recommendation systems provide a mechanism to assist users in classifying users with similar interests
- The collection of the data should be done from very profound sites hence the data is correct and accurate.
- The algorithm designed should be such that the result obtained is optimum.

Chapter 4: Methodology

4.1 Introduction

To build this system we collected sample data from MovieLens website. These data consists of 20 million movie ratings along with user-ids and genres and tags. Our very first step was to visualize the whole data-set. After visualization, the next step was to build the user-item matrix which is a matrix of the ratings a user has given to each movie they have seen, the column here is the user Id and the row here is the movieId.

Upon building this user – item matrix we see that the user-item matrix is a sparse matrix. Now a sparse matrix is a matrix with maximum number of zero values in it. So we check the sparsity of the data.

4.2 Approach used to address the Problem

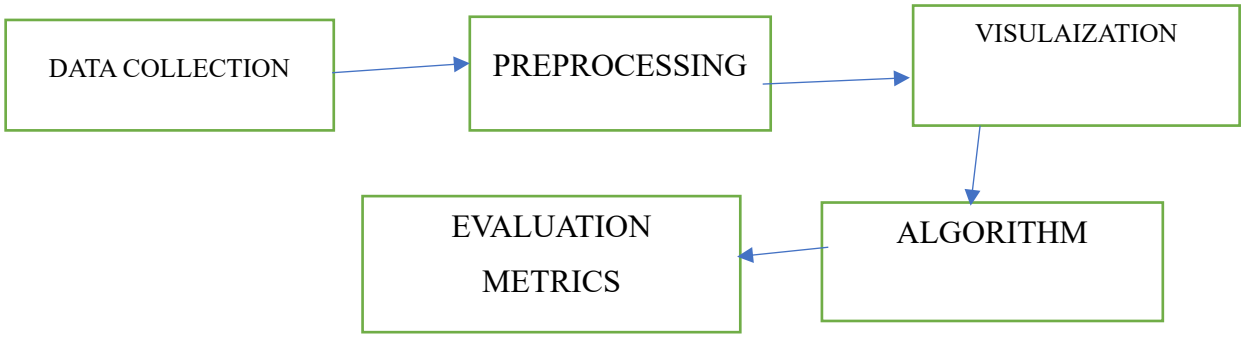


Figure no.1

4.3 Steps/Phases involved

1. Collection of datasets from the profound websites.
2. Pre-processing of the dataset so that it will be available to use in for the following algorithmic operations.
3. Visualizing the actual dataset via plots such as histogram, scatterplot etc.
4. Developing customized algorithm for collaborative filtering to predict ratings matrix.
5. Setting up SVD algorithm to find similar users.
6. Evaluation metrics

4.4 Algorithm Description

The steps to design the algorithm for this problem are given below:

MAX_ROWS,MAX_COLUMNS = matrix.shape

We define a function computeSVD( ) which takes in parameters the user-rating matrix and the dimensional reduction value K:

1. U,s,Vt= sparsesvd(urm,K)

2. dim = (len(s),len(s))

3. S = np.zeros(dim,dtype = np.float32) Loop is set from 0 to len(s) and sqrt() values of s[i] is stored in S[i,i]

4. U=csc_matrix(np.transpose(U),dtype = np.float32) 5. S = csc_matrix(S,dtype = np.float32)

6. Vt = csc_matrix(Vt, dtype = np.float32)

7. return U , S ,Vt

The above function returns the reduced matrix(in decomposed form) . Next We define another function called computeEstimatedRatings() which takes in parameters like the user-rating matrix, U, S, Vt , test_userid, K and a Boolean value test

1.rightTerm=S*Vt

2.estimatedRatings=np.zeros(shape=(MAX_Rows,MAX_Colum ns), dtype=np.float16)

3. for userTest in uTest:

prod = U[userTest, :]*rightTerm we convert the vector to dense format in order to get the indices of the movies with the best estimated ratings

4.estimatedRatings[userTest,:]=prod.todense(recom=(estimatedRatings[userTest,:]).argsort()[: 10]

5. return recom

This function returns the final recommended list of movies based on the input parameters.

4.5 Techniques Used for Analysis

Initially , we used collaborative filtering to fill the gaps of ratings matrix and then customized Single Value Decomposition algorithm will be used to recommend top-N movies to the user of that year. Apart from that we included Speech recognition to collect the interest of the user and then to recommend the movies via speech. Finally, we calculated the evaluation metrics for the model.

Chapter 5: Design and Implementation

5.1 Introduction

In designing the system, for behavioural design we considered class UML diagram as shown in below section. It showcases the relation between different entities and depicts the flow of the program. For detailed explanation, the implementation section briefs about it.

5.2 Design of the System

5.2.1 Behavioural Design

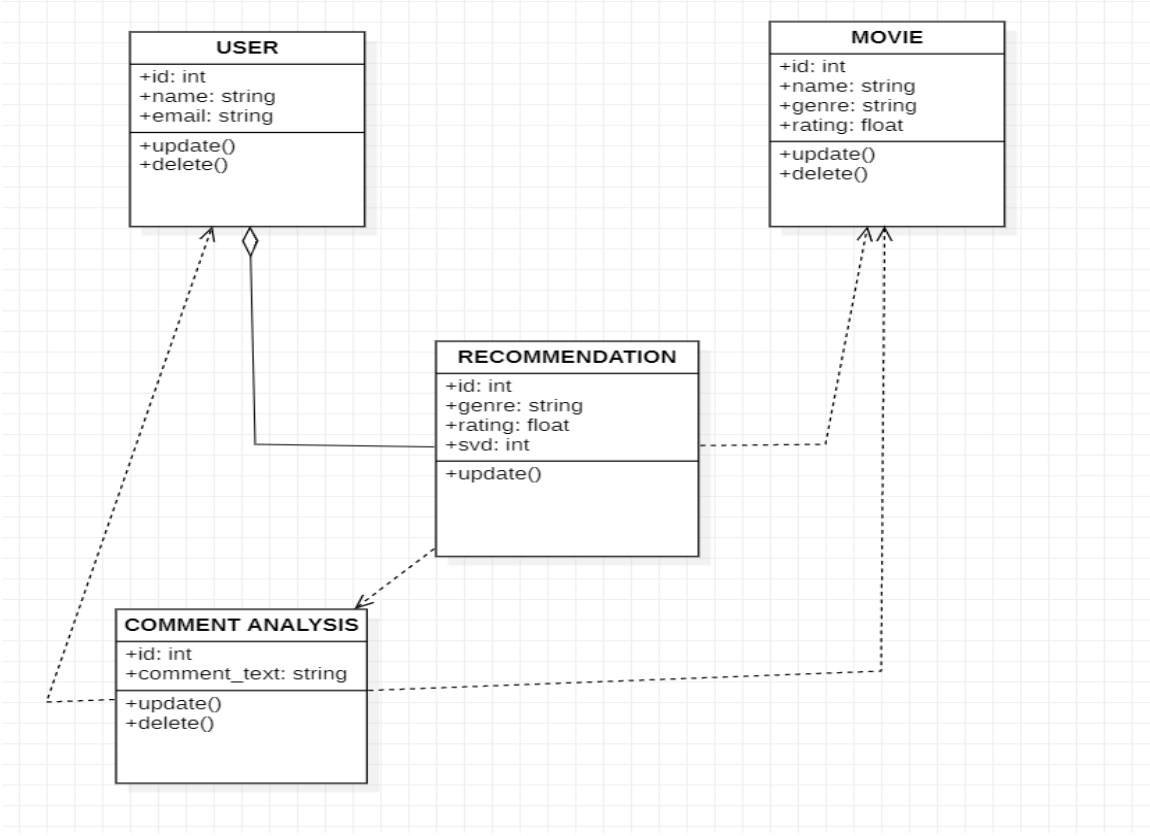


Figure no.2

5.2.2 User Interface Design

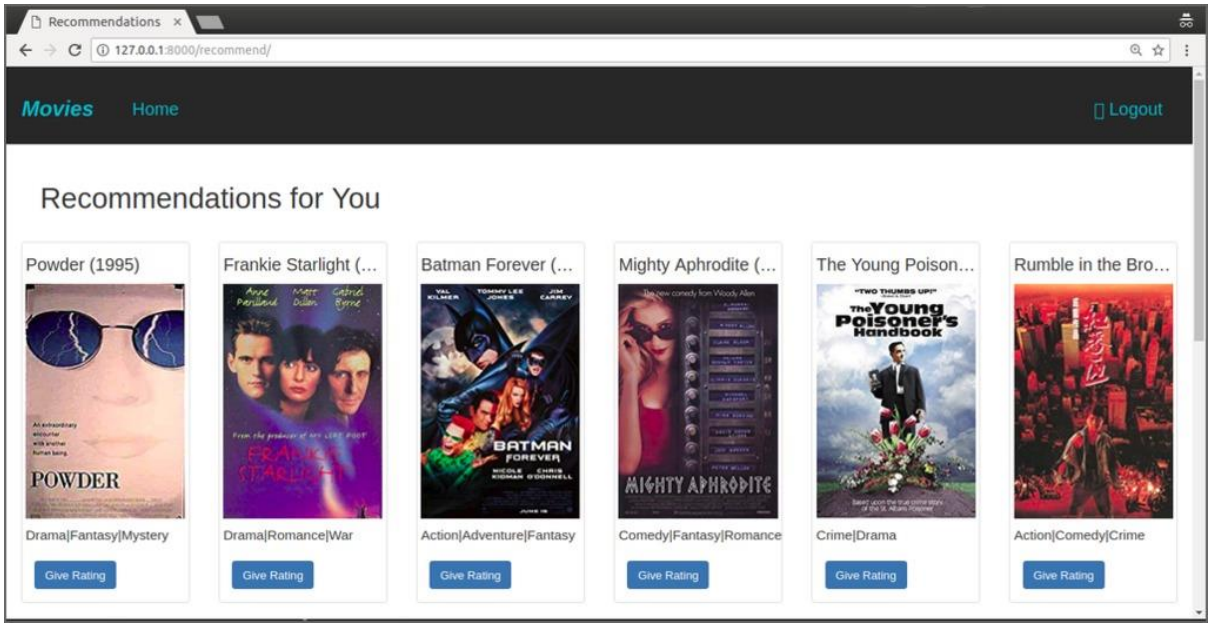


Figure no.3

5.3 Implementation

5.3.1 Dataset

In this we used “movie.csv” and “rating.csv” datasets to extract the data about movies which includes movie name, year of release, comments, rating of a movie given by viewer, journals, actor. The dataset contains a total of 100000 rows to process. Initially, titleId column consists of both the movie and its year of release. UserId, movieId, genre are the other features in the “movie.csv” dataset whereas “rating.csv” dataset consists of userid, movieid, rating, timestamp.

5.3.2 Preprocessing

To train the model first we need to process it to avoid noisy data, to full null values and to delete/replace unnecessary entities which are not needed for our project.

At first, splitting the title and release year in separate columns in movie dataframe. Converting year to timestamp. Categorize movies genres properly through that it can be used to work with +20MM rows string. Modify rating timestamp format from seconds to datetime year. Check and cleaning the NaN values. The above mentioned are preprocessing steps required to built required dataset.

5.3.3 SVD+COLLABORATIVE:

Importing necessary libraries from sklearn and scipy to fill null values, to replace with necessary values and to shape the dataset. They help to extract the year and remove the parentheses and removing the years from the ‘title’ column. Applying the strip function to get rid of any ending whitespace characters that may have appeared.

5.3.4 SPEECH TO TEXT:

Importing speech recognizer to detect the speech and declaring the required conditions whether the google understood the voice or not. If not it throws inbuilt exception. And the recognizer wait a while to adjust the energy threshold based on the surrounding noise level.

5.3.5 TEXT TO SPEECH:

Converting Text to Speech using google text to speech api and initialize the language that we need to convert (i.e English). Passing the text and language to the engine, here we have marked slow=False. Which tells the module that the converted audio should have a high speed. Saving the converted audio in a mp3 file.

5.4 Detailed Description of the Code and Algorithm

5.4.1 Loading libraries and dataset

```
#Loading Dataset and libraries
from math import sqrt
import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import linear_kernel
from timeit import default_timer
movies=pd.read_csv("movied.csv",sep=",",nrows=1000000)
ratings=pd.read_csv("rating.csv",sep=",",nrows=1000000)
```

5.4.2 Data Preprocessing

```
# Split title and release year in separate columns in movies dataframe. Convert year to timestamp.
movies['year'] = movies.title.str.extract("\((\d{4})\)", expand=True)
movies.year = pd.to_datetime(movies.year, format='%Y')
movies.year = movies.year.dt.year # As there are some NaN years, resulting type will be float (decimals)
movies.title = movies.title.str[:-7]

# Categorize movies genres properly. Working later with +20MM rows of strings proved very resource consuming
genres_unique = pd.DataFrame(movies.genres.str.split('|').tolist()).stack().unique()
genres_unique = pd.DataFrame(genres_unique, columns=['genre']) # Format into DataFrame to store later
movies = movies.join(movies.genres.str.get_dummies().astype(bool))
movies.drop('genres', inplace=True, axis=1)

# Modify rating timestamp format (from seconds to datetime year)
#ratings.timestamp = pd.to_datetime(ratings.timestamp, unit='s')
ratings.timestamp = pd.to_datetime(ratings.timestamp, infer_datetime_format=True)
ratings.timestamp = ratings.timestamp.dt.year

# Check and clean NaN values
print ("Number of movies Null values: ", max(movies.isnull().sum()))
print ("Number of ratings Null values: ", max(ratings.isnull().sum()))
movies.dropna(inplace=True)
ratings.dropna(inplace=True)

# Organise a bit, then save into feather-formatand clear from memory
movies.sort_values(by='movieId', inplace=True)
ratings.sort_values(by='movieId', inplace=True)
movies.reset_index(inplace=True, drop=True)
ratings.reset_index(inplace=True, drop=True)
```

5.4.3 COLLABORATIVE FILTERING

```
from sklearn.metrics import pairwise_distances
```

```
from scipy.spatial.distance import cosine, correlation
df_movies_ratings=pd.merge(df_movies, df_ratings)
df_movies_ratings
ratings_matrix_items =
df_movies_ratings.pivot_table(index=['movieId'],columns=['userId'],values='rating').reset_in
dex(drop=True)
ratings_matrix_items.fillna( 0, inplace = True )
ratings_matrix_items.shape
```

5.4.4 SVD

```
from scipy.sparse.linalg import svds
Ratings = ratings.pivot(index = 'userId', columns ='movieId', values = 'rating').fillna(0)
Ratings.head()
R = Ratings.to_numpy()
#print(R)
user_ratings_mean = np.mean(R, axis = 1)
#print(user_ratings_mean.shape)
print(user_ratings_mean.size)
Ratings_demeaned = R - user_ratings_mean.reshape(-1, 1) ## Making the user_ratings_mean
vertical by reshaping
U, sigma, Vt = svds(Ratings_demeaned, k = 50)
print('Size of sigma: ' , sigma.size)
sigma = np.diag(sigma)
print('Shape of sigma: ', sigma.shape)
print(sigma)
print('Shape of U: ', U.shape)
print('Shape of Vt: ', Vt.shape)
all_user_predicted_ratings = np.dot(np.dot(U, sigma), Vt) + user_ratings_mean.reshape(-1, 1)
print('All user predicted rating : ', all_user_predicted_ratings.shape)
preds = pd.DataFrame(all_user_predicted_ratings, columns = Ratings.columns)
preds.head()
```

5.4.5Algorithm:

First, we consider a dataset with movieId, userId, titleId, and other features we preprocess the dataset. And after that we consider collaborative filtering to fill the NaN values of ratings

matrix and create new matrix. Now this ratings matrix is used in SVD algorithms. By defining a recommendation function we customize the output of recommended movies for the user based on ratings and genre. Now we call speech recognition library[14]. Then define speech to text function using this function we can convert speech text and put it in the recommendation function and get the output. Now , we defined another function that is text to speech function using this function we can convert the movies to speech. This recommendation system enables speech recommendation movies.

Chapter 6: Testbed Execution

6.1 Data Set Description

The dataset is taken from Movie lens and specifically movie and ratings datasets are considered from them. Reviews dataset is taken from the IMDB website. The dataset contains a total of 100000 rows to process. Initially, titleId column consists of both the movie and its year of release. UserId, movieId, genre are the other features in the "movie.csv" dataset whereas "rating.csv" dataset consists of userid, movieid, rating, timestamp.

6.2 Execution Steps

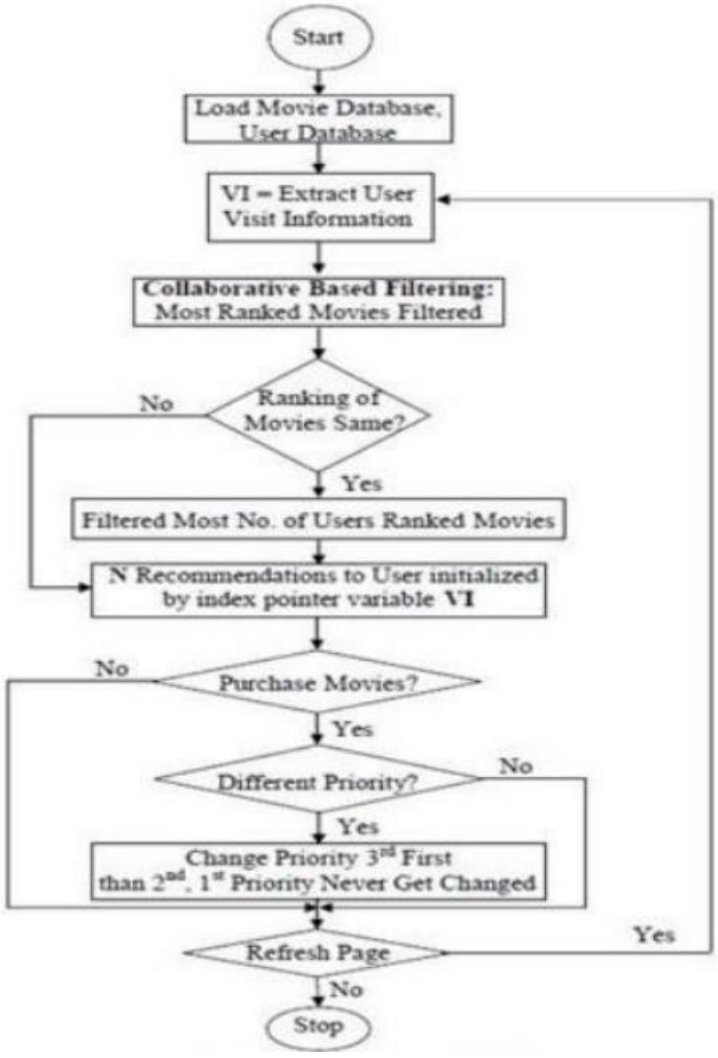


Figure no.4

6.3 Benchmarking Strategy

Recommendation of movies via speech is strategy tool used in this recommendation system. This option is not available for the users in the market platforms like Amazon Prime, Netflix.

Chapter 7: Results and Discussion

7.1 Result Description

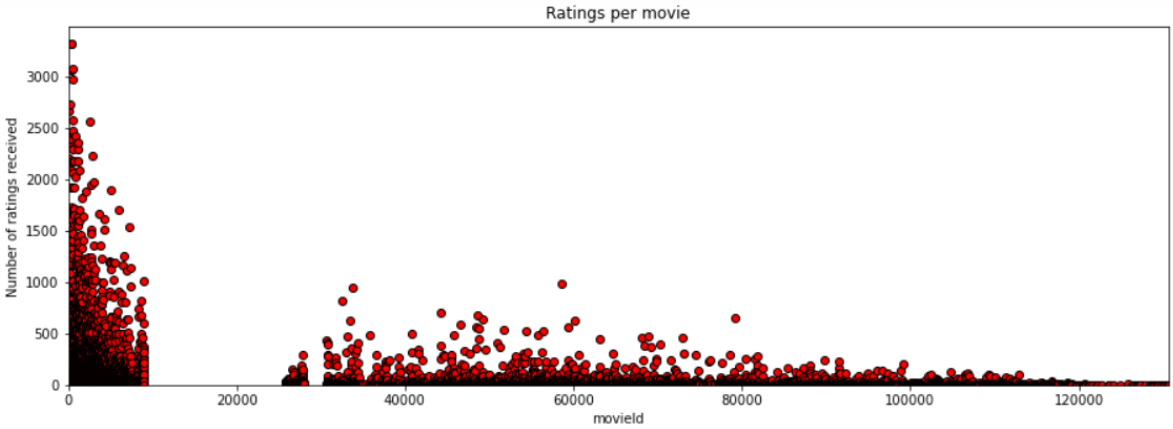


Figure no.5

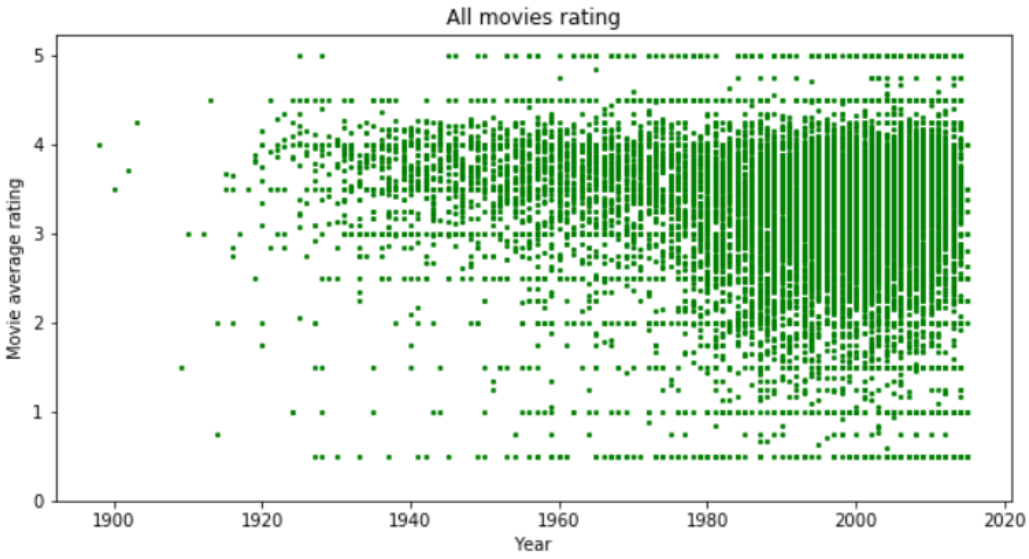


Figure no.6

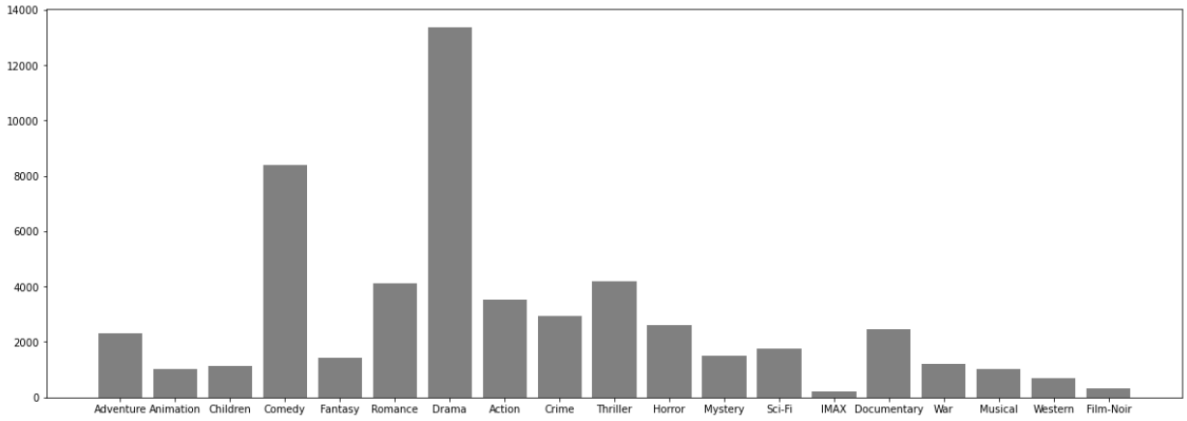


Figure no.7

Collaborative Based Recommendation Technique				
No. of Inputs	Accuracy	Precision	Recall	F-Measure
60	0.79	0.82	0.81	0.8340741
80	0.8	0.81	0.78	0.804031
100	0.81	0.8	0.75	0.7639669
120	0.78	0.77	0.75	0.7439316
140	0.75	0.74	0.68	0.7238938
160	0.76	0.73	0.67	0.7038532
180	0.74	0.7	0.64	0.6737864
200	0.68	0.66	0.6	0.6336842

Figure no.8

SVD +COLLABORATIVE BASED RECOMMENDATION							
	FOLD1	FOLD2	FOLD3	FOLD 4	FOLD 5	MEAN	STD
RMSE	0.9460	0.9616	0.9710	0.9499	0.9573	0.9573	0.0088
MAE	0.7417	0.7475	0.7438	0.7585	0.7438	0.7470	0.0060

Table1

7.2 Analysis of Results

If we look into the visualisation part in the above section , Figure no.5 depicts the individual user id and the number of ratings they have given. In the following graph, it showcseas the connection between movie released year and ratings of the movie by user.

Next table is taken from the standard collaborative recommendation evaluation metrics from research paper. But comparing the results with hybrid approach of recommendation system with CV =5 RMSE is 0.8450 and K-Fold is 0.9710 which is pretty good.

7.3 Interpretation of Results

RMSE and MAE scores implicates the answer for scalability issue and it can also be improved in future.

7.4 Benchmarking the approach

It seems to be usual approach but there is a lot of customization algorithms implemented. Another interesting element of the approach is recommendation via speech. It is not implemented in any of the recommendation system. On comparing our algorithmic approach

with existing one, our system yielded RMSE score 0.9710 whereas when you refer to Figure no.8 its accuracy score did not exceed 0.81.

7.5 Significance and implications for future research.

Significantly this work can be continued for future research in the field of Natural Language Processing. This system works for collecting movies and then recommend movies based on user rating and genre. For extension of this it can collect the user mood and suggest them with the related movie. For example, if user is feeling sad then the system would recommend comedy or motivational movies based on his previous ratings.

Chapter 8: SCREENSHOTS

In [1]:
```
#Loading Dataset and libraries
```

In [2]:
```
from math import sqrt
import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import linear_kernel
from timeit import default_timer
```

In [22]:
```
movies=pd.read_csv("movied.csv",sep=",",nrows=1000000)
ratings=pd.read_csv("rating.csv",sep=",",nrows=1000000)
```

Type *Markdown* and LaTeX: α^2

In [23]:
```
df_movies = movies
df_ratings = ratings
print("MOVIE\n",df_movies.head(5))
print("RATINGS\n",df_ratings.head(5))
```

```
MOVIE
    movieId                               title  \
0        1                    Toy Story (1995)
1        2                      Jumanji (1995)
2        3             Grumpier Old Men (1995)
3        4            Waiting to Exhale (1995)
4        5  Father of the Bride Part II (1995)

                                        genres
0  Adventure|Animation|Children|Comedy|Fantasy
1                   Adventure|Children|Fantasy
2                               Comedy|Romance
```

```
RATINGS
    userId  movieId  rating            timestamp
0       1        2     3.5  2005-04-02 23:53:47
1       1       29     3.5  2005-04-02 23:31:16
2       1       32     3.5  2005-04-02 23:33:39
3       1       47     3.5  2005-04-02 23:32:07
4       1       50     3.5  2005-04-02 23:29:40
```

n [5]:
```
# Split title and release year in separate columns in movies dataframe. Convert year to timestamp.
movies['year'] = movies.title.str.extract("\((\d{4})\)", expand=True)
movies.year = pd.to_datetime(movies.year, format='%Y')
movies.year = movies.year.dt.year # As there are some NaN years, resulting type will be float (decimals)
movies.title = movies.title.str[:-7]

# Categorize movies genres properly. Working later with +20MM rows of strings proved very resource consuming
genres_unique = pd.DataFrame(movies.genres.str.split('|').tolist()).stack().unique()
genres_unique = pd.DataFrame(genres_unique, columns=['genre']) # Format into DataFrame to store later
movies = movies.join(movies.genres.str.get_dummies().astype(bool))
movies.drop('genres', inplace=True, axis=1)

# Modify rating timestamp format (from seconds to datetime year)
#ratings.timestamp = pd.to_datetime(ratings.timestamp, unit='s')
ratings.timestamp = pd.to_datetime(ratings.timestamp, infer_datetime_format=True)
ratings.timestamp = ratings.timestamp.dt.year

# Check and clean NaN values
print ("Number of movies Null values: ", max(movies.isnull().sum()))
print ("Number of ratings Null values: ", max(ratings.isnull().sum()))
movies.dropna(inplace=True)
ratings.dropna(inplace=True)

# Organise a bit, then save into feather-formatand clear from memory
movies.sort_values(by='movieId', inplace=True)
ratings.sort_values(by='movieId', inplace=True)
movies.reset_index(inplace=True, drop=True)
ratings.reset_index(inplace=True, drop=True)
```

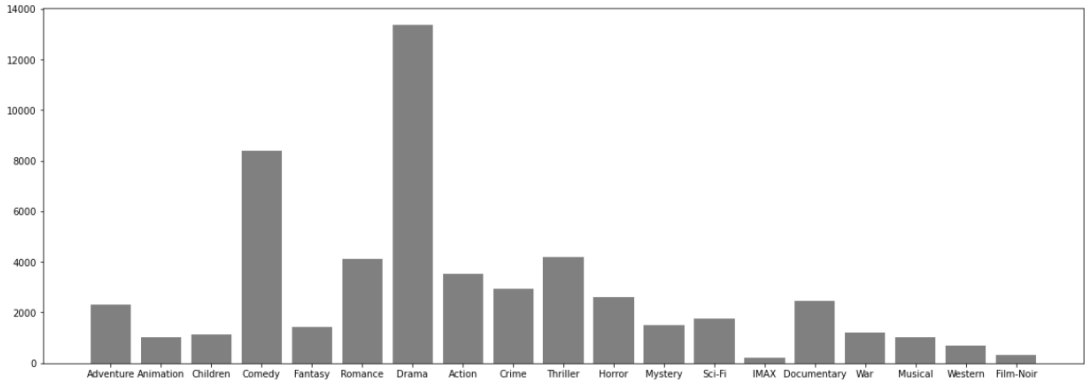
```
Number of movies Null values:  22
Number of ratings Null values:  0
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 27256 entries, 0 to 27255
Data columns (total 23 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   movieId             27256 non-null  int64
 1   title               27256 non-null  object
 2   year                27256 non-null  float64
 3   (no genres listed)  27256 non-null  bool
 4   Action              27256 non-null  bool
 5   Adventure           27256 non-null  bool
 6   Animation           27256 non-null  bool
 7   Children            27256 non-null  bool
 8   Comedy              27256 non-null  bool
 9   Crime               27256 non-null  bool
 10  Documentary         27256 non-null  bool
```

In [25]:
```
plt.figure(figsize=(20,7))
generlist = df_movies['genres'].apply(lambda generlist_movie : str(generlist_movie).split("|"))
geners_count = {}

for generlist_movie in generlist:
    for gener in generlist_movie:
        if(geners_count.get(gener,False)):
            geners_count[gener]=geners_count[gener]+1
        else:
            geners_count[gener] = 1
geners_count.pop("(no genres listed)")
plt.bar(geners_count.keys(),geners_count.values(), color = 'grey')
```

Out[25]: <BarContainer object of 19 artists>



In [8]:
```
#Collaborative filtering
```

In [9]:
```
from sklearn.metrics import pairwise_distances
from scipy.spatial.distance import cosine, correlation
```

In [10]:
```
df_movies_ratings=pd.merge(df_movies, df_ratings)
```

In [11]:
```
df_movies_ratings
```

In [26]:
```
df_movies_ratings
```

Out[26]:

	movieId	title	genres	year	userId	rating	timestamp
0	1	Toy Story	Adventure|Animation|Children|Comedy|Fantasy	1995.0	1110	4.0	2000
1	1	Toy Story	Adventure|Animation|Children|Comedy|Fantasy	1995.0	2038	4.5	2004
2	1	Toy Story	Adventure|Animation|Children|Comedy|Fantasy	1995.0	2037	5.0	1998
3	1	Toy Story	Adventure|Animation|Children|Comedy|Fantasy	1995.0	2036	5.0	2001
4	1	Toy Story	Adventure|Animation|Children|Comedy|Fantasy	1995.0	5178	3.5	2006
...	...	...	...	...	...	...	...
999995	130219	The Dark Knight	Action|Crime|Drama|Thriller	2011.0	1339	4.5	2015
999996	130462	The Boy	(no genres listed)	2015.0	5731	4.0	2015
999997	130490	Insurgent	Action|Romance|Sci-Fi	2015.0	2423	1.0	2015
999998	130490	Insurgent	Action|Romance|Sci-Fi	2015.0	3397	3.5	2015
999999	130642	Backcountry	Drama|Horror|Thriller	2014.0	3858	3.0	2015

1000000 rows × 7 columns

In [12]:
```
ratings_matrix_items = df_movies_ratings.pivot_table(index=['movieId'],columns=['userId'],values='rating').reset_index(drop=True)
ratings_matrix_items.fillna( 0, inplace = True )
ratings_matrix_items.shape
```

Out[12]: (13950, 6743)

In [28]:
```
ratings_matrix_items
```

Out[28]:

userId	1	2	3	4	5	6	7	8	9	10	...	6734	6735	6736	6737	6738	6739	6740	6741	6742	6743
0	0.0	0.0	4.0	0.0	0.0	5.0	0.0	4.0	0.0	4.0	...	0.5	4.0	4.0	3.0	0.0	4.0	0.0	4.0	0.0	4.0
1	3.5	0.0	0.0	0.0	3.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.0
2	0.0	4.0	0.0	0.0	0.0	3.0	3.0	5.0	0.0	0.0	...	0.0	5.0	0.0	0.0	0.0	4.0	0.0	0.0	0.0	3.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	4.0	3.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
13945	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
13946	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
13947	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
13948	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
13949	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

13950 rows × 6743 columns

In [14]:
```
#Setting Up SVD
```

In [15]:
```
from scipy.sparse.linalg import svds
```

In [16]:
```
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())
```

```
C:\Users\91799\AppData\Local\Temp\ipykernel_43748\4290400506.py:5: FutureWarning: The default value of regex will change from T
rue to False in a future version.
  movies['title'] = movies.title.str.replace('(\(\d\d\d\d\))', '')
```

In [17]:
```
Ratings = ratings.pivot(index = 'userId', columns ='movieId', values = 'rating').fillna(0)
Ratings.head()
R = Ratings.to_numpy()
#print(R)
user_ratings_mean = np.mean(R, axis = 1)
#print(user_ratings_mean.shape)
print(user_ratings_mean.size)
Ratings_demeaned = R - user_ratings_mean.reshape(-1, 1) ## Making the user_ratings_mean vertical by reshaping
```

```
6743
```

In [18]:
```
U, sigma, Vt = svds(Ratings_demeaned, k = 50)
print('Size of sigma: ' , sigma.size)
```

```
Size of sigma:  50
```

In [19]:
```
sigma = np.diag(sigma)
print('Shape of sigma: ', sigma.shape)
print(sigma)
```

```
print('Shape of U: ', U.shape)
print('Shape of Vt: ', Vt.shape)
```

```
Shape of sigma:  (50, 50)
[[ 142.61249165    0.            0.         ...    0.
     0.            0.        ]
 [   0.          143.85301215    0.         ...    0.
     0.            0.        ]
 [   0.            0.          144.52622401 ...    0.
     0.            0.        ]
 ...
 [   0.            0.            0.         ...  614.79347143
     0.            0.        ]
 [   0.            0.            0.         ...    0.
   656.79707909    0.        ]
 [   0.            0.            0.         ...    0.
     0.         1560.07209886]]
Shape of U:  (6743, 50)
Shape of Vt:  (50, 13950)
```

7]:
```
all_user_predicted_ratings = np.dot(np.dot(U, sigma), Vt) + user_ratings_mean.reshape(-1, 1)
print('All user predicted rating : ', all_user_predicted_ratings.shape)
preds = pd.DataFrame(all_user_predicted_ratings, columns = Ratings.columns)
```

In [34]:
```
preds.head()
```

```
All user predicted rating :  (6743, 13950)
```

Out[34]:

movieId	1	2	3	4	5	6	7	8	9	10	...	129350	129354	129428	129707
0	0.462816	0.886560	0.116673	0.008308	-0.215565	0.110520	-0.358674	-0.024280	-0.217334	0.032192	...	0.007858	-0.009063	-0.000590	-0.002358
1	0.979778	0.101947	0.331502	0.084563	0.198166	0.225804	0.464224	-0.000231	0.089829	-0.063064	...	-0.002574	-0.007137	-0.005901	-0.000340
2	2.042083	0.828816	-0.159202	-0.011740	-0.123303	0.469264	0.055829	0.002462	-0.053686	0.352812	...	0.003774	0.014084	0.007017	0.004093
3	-0.527892	0.673756	0.420112	0.001046	0.276751	0.840723	0.090925	0.090294	0.186957	0.946194	...	0.001692	0.001809	0.001136	0.002849
4	2.246131	1.121891	1.234257	0.115929	1.183925	0.619121	1.367333	0.157426	0.296183	1.433395	...	-0.000493	0.009807	-0.009911	0.001258

5 rows × 13950 columns

In [21]:
```
def recommend_movies(predictions, userID, movies, original_ratings, num_recommendations):
    """
    Implementation of SVD by hand
    :param predictions : The SVD reconstructed matrix,
    userID : UserId for which you want to predict the top rated movies,
    movies : Matrix with movie data, original_ratings : Original Rating matrix,
    num_recommendations : num of recos to be returned
    :return: num_recommendations top movies
    """
    # Get and sort the user's predictions
    user_row_number = userID - 1 # User ID starts at 1, not 0
    sorted_user_predictions = predictions.iloc[user_row_number].sort_values(ascending=False) # User ID starts at 1

    # Get the user's data and merge in the movie information.
    user_data = original_ratings[original_ratings.userId == (userID)]
    user_full = (user_data.merge(movies, how = 'left', left_on = 'movieId', right_on = 'movieId').
                    sort_values(['rating'], ascending=False)
                )

    print('User {0} has already rated {1} movies.'.format(userID, user_full.shape[0]))
    print('Recommending highest {0} predicted ratings movies not already rated.'.format(num_recommendations))

    # Recommend the highest predicted rating movies that the user hasn't seen yet.
    recommendations = (movies[~movies['movieId'].isin(user_full['movieId'])].
         merge(pd.DataFrame(sorted_user_predictions).reset_index(), how = 'left',
               left_on = 'movieId',
               right_on = 'movieId').
         rename(columns = {user_row_number: 'Predictions'}).
         sort_values('Predictions', ascending = False).
                       iloc[:num_recommendations, :-1]
                      )

    return user_full, recommendations
```

In [31]:
```
# Define a TF-IDF Vectorizer Object.
tfidf_movies_genres = TfidfVectorizer(token_pattern = '[a-zA-Z0-9\-]+')

#Replace NaN with an empty string
df_movies['genres'] = df_movies['genres'].replace(to_replace="(no genres listed)", value="")

#Construct the required TF-IDF matrix by fitting and transforming the data
tfidf_movies_genres_matrix = tfidf_movies_genres.fit_transform(df_movies['genres'])
# print(tfidf_movies_genres.get_feature_names())
# Compute the cosine similarity matrix
# print(tfidf_movies_genres_matrix.shape)
# print(tfidf_movies_genres_matrix.dtype)
cosine_sim_movies = linear_kernel(tfidf_movies_genres_matrix, tfidf_movies_genres_matrix)
# print(cosine_sim_movies)
def get_recommendations_based_on_genres(movie_title, cosine_sim_movies=cosine_sim_movies):
    """
    Calculates top 2 movies to recommend based on given movie titles genres.
    :param movie_title: title of movie to be taken for base of recommendation
    :param cosine_sim_movies: cosine similarity between movies
    :return: Titles of movies recommended to user
    """
    #df_movies['title'] = df_movies['title'].fillna("").astype('str')
    # Get the index of the movie that matches the title
    idx_movie = df_movies.loc[df_movies['title'].isin([movie_title])]
    idx_movie = idx_movie.index

    # Get the pairwsie similarity scores of all movies with that movie
    sim_scores_movies = list(enumerate(cosine_sim_movies[idx_movie][0]))

    # Sort the movies based on the similarity scores
    sim_scores_movies = sorted(sim_scores_movies, key=lambda x: x[1], reverse=True)

    # Get the scores of the 10 most similar movies
    sim_scores_movies = sim_scores_movies[1:3]

    # Get the movie indices
    movie_indices = [i[0] for i in sim_scores_movies]
```

In [32]:
```
#Converting the text to speech using google text to speech api
import random
def text_to_speech():
    file = open('dd.txt','w')
    file.writelines(f'hello User \n')
    file.writelines('The Movie You Should Watched Next are \n')
    for res in result:
        res = ''.join(res)
        file.write(res+'\n')
    file.close()
    file = open('dd.txt','r')
    data= file.read()
    print(data)
    file.close()


    # Language in which you want to convert
    language = 'en'

    # Passing the text and language to the engine,
    # here we have marked slow=False. Which tells
    # the module that the converted audio should
    # have a high speed
    myobj = gtts.gTTS(text=data, lang=language, slow=False)

    # Saving the converted audio in a mp3 file named
    i=random.randint(1,100)
    file='new'+str(i)+'.mp3'
    myobj.save(file)

    #Playing the converted file
    playsound(file)
```

In [34]:
```
import speech_recognition as sr
def speech_to_text():

    sample_rate = 48000

    chunk_size = 2048
    #Initialize the recognizer
    r = sr.Recognizer()

    with sr.Microphone(sample_rate = sample_rate,
                       chunk_size = chunk_size) as source:
        #wait for a second to let the recognizer adjust the
        #energy threshold based on the surrounding noise level
        r.adjust_for_ambient_noise(source)
        print ("Speak the name of the movie ")
        #listens for the user's input
        audio = r.listen(source)

        try:
            text = r.recognize_google(audio)
            print(text)

        #error occurs when google could not understand what was said

        except sr.UnknownValueError:
            print("Google Speech Recognition could not understand audio")

        except sr.RequestError as e:
            print("Could not request results from Google  Speech Recognition service; {0}".format(e))

        return text
```

In [46]:
```
from playsound import playsound
import re
import gtts
text=speech_to_text()
result=get_recommendations_based_on_genres(text)
text_to_speech()
```

```
Speak the name of the movie
Toy Story
hello User
The Movie You Should Watched Next are
Antz
Toy Story 2
```

In [37]:
```
# Top 20 movies that User 150 has rated
already_rated.head(20)
```

Out[37]:

	userId	movieId	rating	timestamp	title	year	(no genres listed)	Action	Adventure	Animation	...	Film-Noir	Horror	IMAX	Musical	Mystery	Romance	Sci-Fi	Th
25	150	66934	5.0	2010	Dr. Horrible's Sing-Along Blog	NaN	False	False	False	False	...	False	False	False	True	False	False	True	
24	150	7153	4.0	2010	Lord of the Rings: The Return of the King, The	NaN	False	True	True	False	...	False	False	False	False	False	False	False	
23	150	5952	4.0	2010	Lord of the Rings: The Two Towers, The	NaN	False	False	True	False	...	False	False	False	False	False	False	False	
22	150	4993	4.0	2010	Lord of the Rings: The Fellowship of the Ring,...	NaN	False	False	True	False	...	False	False	False	False	False	False	False	
6	150	1033	4.0	2010	Fox and the Hound, The	NaN	False	False	False	True	...	False	False	False	False	False	False	False	
7	150	1298	4.0	2010	Pink Floyd: The Wall	NaN	False	False	False	False	...	False	False	False	True	False	False	False	
9	150	1586	4.0	2010	G.I. Jane	NaN	False	True	False	False	...	False	False	False	False	False	False	False	
20	150	2926	4.0	2010	Hairspray	NaN	False	False	False	False	...	False	False	False	False	False	False	False	
12	150	2114	4.0	2010	Outsiders, The	NaN	False	False	False	False	...	False	False	False	False	False	False	False	

Out[38]:

	movieId	title	year	(no genres listed)	Action	Adventure	Animation	Children	Comedy	Crime	...	Film-Noir	Horror	IMAX	Musical	Mystery	Romance	Sci-Fi
2467	2571	Matrix, The	NaN	False	True	False	False	False	False	False	...	False	False	False	False	False	False	True
2852	2959	Fight Club	NaN	False	True	False	False	False	False	True	...	False	False	False	False	False	False	False
1165	1197	Princess Bride, The	NaN	False	True	True	False	False	True	False	...	False	False	False	False	False	True	False
6405	6539	Pirates of the Caribbean: The Curse of the Bla...	NaN	False	True	True	False	False	True	False	...	False	False	False	False	False	False	False
2657	2762	Sixth Sense, The	NaN	False	False	False	False	False	False	False	...	False	True	False	False	True	False	False
12499	58559	Dark Knight, The	NaN	False	True	False	False	False	False	True	...	False	False	True	False	False	False	False
107	110	Braveheart	NaN	False	True	False	False	False	False	False	...	False	False	False	False	False	False	False
1106	1136	Monty Python and the Holy Grail	NaN	False	False	True	False	False	True	False	...	False	False	False	False	False	False	False
3466	3578	Gladiator	NaN	False	True	True	False	False	False	False	...	False	False	False	False	False	False	False
4190	4306	Shrek	NaN	False	False	True	True	True	True	False	...	False	False	False	False	False	True	False
2578	2683	Austin Powers: The Spy Who Shagged Me	NaN	False	True	True	False	False	True	False	...	False	False	False	False	False	False	False
		Amelie (Fabuleux																

CONCLUSION

Recommender systems open new opportunities of retrieving personalized information on the Internet. It also helps to alleviate the problem of information overload which is a very common phenomenon with information retrieval systems and enables users to have access to products and services which are not readily available to users on the system. We come up with a strategy that focuses on dealing with user's personal interests and based on his previous reviews, movies are recommended to users. This strategy helps in improving accuracy of the recommendations. A personal profile can be created for each user, where each user has access to his own history, his likes, ratings, comments, password modification processes. It also helps in collecting authentic data with improved accuracy and makes the system more responsive.

REFERENCES

[1]. Joseph A. Konstan, John Riedl. Recommender systems:from algorithms to user experience. User Model User-AdaptInteract, March 2012

[2]. Chenguang Pan, Wenxin Li. Research Paper Recommendation with Topic Analysis. In Computer Design and Applications IEEE, 2010

[3]. Pu P, Chen L, Hu R. A User-Centric Evaluation Framework for Recommender Systems. In: Proceedings of the fifth ACM conference on Recommender Systems, ACM, 2011

[4]. Bhavik Pathak , Robert Garfinkel , Ram D. Gopal , Rajkumar Venkatesan & Fang Yin .Empirical Analysis of the Impact of Recommender Systems on Sales, Journal of Management Information Systems, December 2014

[5]. Al Mamunur Rashid, Istvan Albert, Dan Cosley, Shyong K. Lam, Sean M. McNee, Joseph A. Konstan, John Riedl. Getting to Know You: Learning New User Preferences in Recommender Systems. In: Proceedings of the international conference on intelligent user interfaces, 2012

[6]. J. Ben Schafer, Joseph Konstan, John Riedl. Recommender Systems in E-Commerce. In: Proceedings of the 1st ACM conference on electronic commerce, 2014.

[7]. A. Merve Acilar, Ahmet Arslan. A collaborative filtering method based on artificial immune network. Expert Systems with Application, 2019

[8]. Chen LS, Hsu FH, Chen MC, Hsu YC. Developing recommender systems with the consideration of product profitability for sellers. Int J Inform Sci 2018.

[9]. Jalali M, Mustapha N, Sulaiman M, Mamay A. WebPUM: A Web-based recommendation system to predict user future movements.Exp Syst Applicat, March 2015.

[10]. Fadhel Aljunid, Mohammed & D H, Manjaiah. (2018). Movie Recommender System Based on Collaborative Filter

[11]. https://scikit-learn.org/

[12]. https://scipy.org/

[13].https://www.thepythoncode.com/article/using-speech-recognition-to-convert-speech-to-text-python

[14]. https://pypi.org/project/SpeechRecognition/

EFFECTIVE RECOMMENDATION SYSTEM

B V N Sravya
SCOPE
M.Tech(Int.), CSE
Vellore Institute Of Technology
sravyabvn29@gmail.com

R Venkata Tagore Reddy
SCOPE
M.Tech(Int.), CSE
Vellore Institute Of Technology
rvtagorereddy@gmail.com

D Nithin Sri Sarveswar
SCOPE
M.Tech(Int.), CSE
Vellore Institute Of Technology
nithinsrisarveswar@gmail.com

Suda Naveen
SCOPE
MTech(Int.),CSE
Vellore Institute Of Technology
sudanaveen9@gmail.com

***Abstract*— This project proposes Recommender systems are efficient tools for filtering online information, which is widespread owing to the changing habits of computer users, personalization trends, and emerging access to the internet. Even though the recent recommender systems are eminent in giving precise recommendations, they suffer from various limitations and challenges like scalability, cold-start, sparsity, etc. Due to the existence of various techniques, the selection of techniques becomes a complex work while building application-focused recommender systems. In addition, each technique comes with its own set of features, advantages and disadvantages which raises even more questions, which should be addressed. This paper aims to undergo a systematic review on various recent contributions in the domain of recommender systems, focusing on diverse applications like books, movies, products, etc. Initially, the various applications of each recommender system are analyzed. Then, the algorithmic analysis on various recommender systems is performed and a taxonomy is framed that accounts for various components required for developing an effective recommender system. In addition, the datasets gathered, simulation platform, and performance metrics focused on each contribution are evaluated and noted. Finally, this review provides a much-needed overview of the current state of research in this field and points out the existing gaps and challenges to help posterity in developing an efficient recommender system.**

Keywords-Recommendation system, hybrid approach

INTRODUCTION

Recommender systems primarily aim to reduce the user's effort and time required for searching relevant information over the internet User's profile (with relevant personal information as well as search history of a user) is compared with search history of different users for a few attributes of the item user is looking for and on the basis of 'ratings' or 'preferences' given by different users for that item, recommender system predicts the recommendations for that item[10]. The filtering techniques have been categorized as 1) Content-based, 2) Collaborative and 3) Hybrid. These are discussed below:

A. The Content-based Approach

Content-based filtering technique works with profiles of users. A profile has information about a user and his preferences. Preferences rely on how the user rated items and what item user has bought and viewed[1]. Generally, once profile is made, recommender systems create a survey, to get basic information about a user so as to avoid the new-user problem[3]. New-user problem arises when profile of the same user is made having different attributes. Profiles are obtained by analyzing the items previously seen and rated by the user and are typically created using keyword analysis techniques from information retrieval[5]. Within

the recommendation method, the engine compares the things that were already positively rated by the user

B. The Collaborative Approach

In collaborative technique, the concept of neighborhood is used. Here, neighbors are the group of people who have rated same item and have similar taste[7]. Items are recommended which have not been rated by the user but were already positively rated by users in his neighborhood that didn't rate and appears for similarities[2].

We can distinguish the collaborative filtering on basis of following famous approaches: i) user-based ii) item-based.

i. User-based approach

The main role is performed by user in the user-based approach. Items will be recommended to the user based on what is viewed by other user from the same group[4]. Recommendations are given to the user based on evaluation of items by other users from the same community, with whom user shares common preferences[9]. If the community rates the item positively, it will be suggested to the user. Thus in the user-based approach the items that were already rated by the user plays an important role in searching a group that shares preferences with him.

ii. Item-based approach

It is fact that the preferences of users remain constant or change very slightly so the similar items build neighborhoods based on appreciations of users. Afterwards the system generates recommendations of items in the neighborhood that a user will prefer[6].

C The Hybrid Approach

This approach combines the multiple filtering methods to get a refined result. Different criteria to combine different recommendation filtering techniques are classified as:

1. Separately implementing content-based methods and merging their predictions

2. Integrate some content-based characteristics into a collaborative approach

3. Integrate some collaborative characteristics into a content-based approach, and

4. Developing a general integrated model that merges characteristics of both content-based and collaborative filtering methods. It reduces the filtering problems as advantages of one technique can be used to minimize the disadvantage of other technique[8].

I. LITERATURE SURVEY

Recommendation systems facilitate users by using supplying useful suggestions, consequently lowering their seek time[9]. The consumer rankings are used for classifying information into numerous classes that could in addition be useful to generate hints[1]. In this, we're going to use statistics mining techniques to examine consumer preferences and determine consumer-unique film ratings via the help of information mining techniques. We can use a film database from IMDB and decide user particular scores for each of them[3]. The analysis of attributes of those films will assist us to become aware of the decisive factors and discover consumer alternatives appropriately[2]. This project introduces a recommendation algorithm based on LSTM and applies it to the recommendation of movies by mining user behavior data and recommending movies with higher ratings to them[4]. This article used the data provided by the movie website Movie Lens. It is the testing set and training set that the data is divided into, and the Top-N recommendation list is produced for the training set, while the algorithm is evaluated on the testing set[7]. It is the features of the data that LSTM can effectively extract and complete the recommendation from the results. According to this paper, the modern optimization for traditional recommendation systems could be summarized as follows[6]. To strengthen user control. Most existing recommendation system according to preset automatically generates multiple recommended users' personal information and requirements, to some extent, limits user participation and control[5]. The result of the

recommendation system should allow users to participate in parameter definition. Recommendation system can be achieved by the relevance feedback mechanism to update the user's real-time demand, for example, by the user to explicitly to the evaluation of the recommended collect user feedback information. Recommender systems use algorithms to provide users with useful recommendations for products or services. Recently, these systems have been using machine learning algorithms from the field of artificial intelligence[9]. Recommendation system is categorized in four classes: Simple System using popularity, Collaborative Filtering, Content based and hybrid based Approach. A movie recommendation is being highly recognized as a part of our social life due to its strength in providing enhanced entertainment[8]. Such a system will predict what movies a user will like based on various criterias like the attributes of previously liked movies by that user or the popularity of the movies. Although a set of movie recommendation systems have been proposed and implemented[10].

PROBLEM DEFINITION:

The problem definition is trying to forecast the opinion the users will have on the dissimilar substance and be able to recommend the finest items to each user.

METHODOLOGY:

To build this system we collected sample data from MovieLens website. These data consists of 20 million movie ratings along with user-ids and genres and tags. Our very first step was to visualize the whole data-set. After visualization, the next step was to build the user-item matrix which is a matrix of the ratings a user has given to each movie they have seen, the column here is the userId and the row here is the movieId.

Upon building this user – item matrix we see that the user-item matrix is a sparse matrix. Now a sparse matrix is a matrix with maximum number of zero values in it. So, we check the sparsity of the data.

The steps to design the algorithm for this problem are given below:

MAX_ROWS, MAX_COLUMNS = matrix.shape

We define a function computeSVD() which takes in parameters the user-rating matrix and the dimensional reduction value K:

1. U,s,Vt= sparsesvd(urm,K)

2. dim = (len(s),len(s))

3. S = np.zeros(dim,dtype = np.float32) Loop is set from 0 to len(s) and sqrt() values of s[i] is stored in S[i,i]

4. U=csc_matrix(np.transpose(U),dtype = np.float32)
5. S = csc_matrix(S,dtype = np.float32)

6. Vt = csc_matrix(Vt, dtype = np.float32)

7. return U , S ,Vt The above function returns the reduced matrix(in decomposed form)

Next We define another function called computeEstimatedRatings() which takes in parameters like the user-rating matrix, U, S, Vt , test_userid, K and a Boolean value test

1.rightTerm=S*Vt
2.estimatedRatings=np.zeros(shape=(MAX_Rows,MAX_Colum ns), dtype=np.float16)

3. for userTest in uTest:

prod = U[userTest, :]*rightTerm we convert the vector to dense format in order to get the indices of the movies with the best estimated ratings

4.estimatedRatings[userTest,:]= prod.todense(recom= (estimatedRatings[userTest,:]).argsort()[:10]

5. return recom This function returns the final recommended list of movies based on the input parameters given by users.

Singular value decomposition (SVD) is a collaborative filtering method for movie recommendation. The aim for the code implementation is to provide users with movies' recommendation from the latent features of item-user matrices. The code would show you how to use the SVD latent factor model for matrix factorization.

A key functionality of a movie recommendation system that uses collaborative filtering and SVD is personalization, which involves tailoring recommendations to each individual user based on their preferences, viewing history, and other factors. This can be done using machine learning algorithms that learn from user behavior over time.

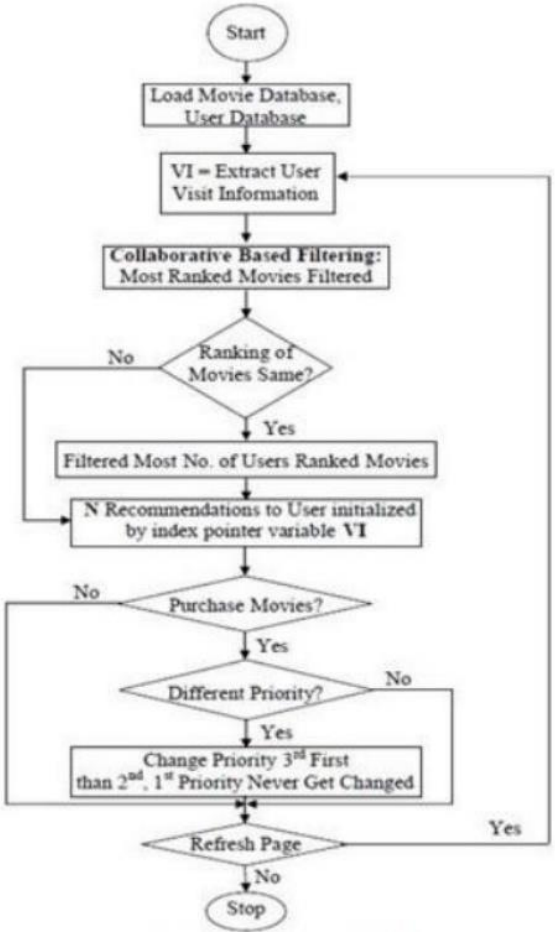


Fig. No.: 1 - Flow chart of SVD+Collaborative

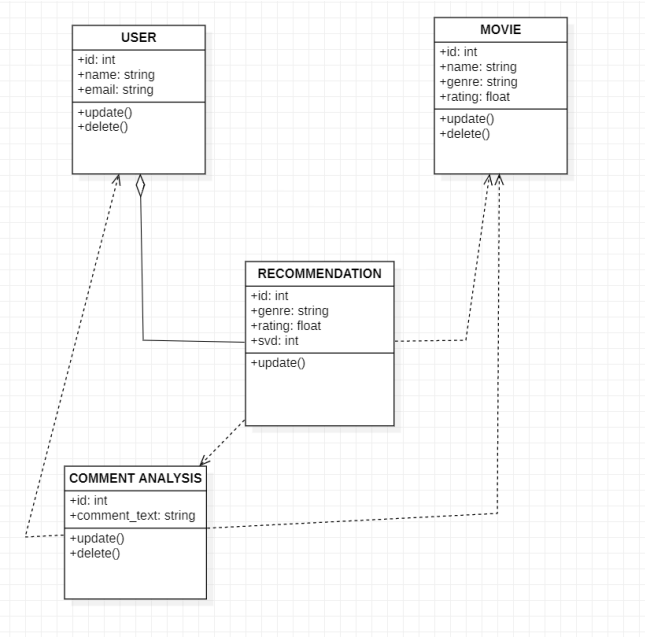


Fig. No: 2 -Structural diagram of Recommendation System.

IMPLEMENTATION:

Dataset Used:

In this we used "movie.csv" and "rating.csv" datasets to extract the data about movies which includes movie name, year of release, comments, rating of a movie given by viewer, journals, actor. The dataset contains a total of 100000 rows to process. In which "movie.csv" contains – movieId, title and genres. Whereas in "rating.csv" contains – userId, movieId, rating, timestamp.

Preprocessing:

To train the model first we need to process it to avoid noisy data, to full null values and to delete/replace unnecessary entities which are not needed for our project.

At first, splitting the title and release year in separate columns in movie dataframe. Converting year to timestamp. Categorize movies genres properly through that it can be used to work with +20MM rows string. Modify rating timestamp format from seconds to datetime year. Check and cleaning the NaN values. The above mentioned are preprocessing steps required to built required dataset.

SVD+Collaborative:

Importing necessary libraries from sklearn and scipy to fill null values, to replace with necessary values and to shape the dataset. They help to extract the year and remove the parentheses and removing the years from the 'title' column. Applying the strip function to get rid of any ending whitespace characters that may have appeared.

Speech to Text:

Importing speech recognizer to detect the speech and declaring the required conditions whether the google understood the voice or not. If not it throws inbuilt exception. And the recognizer wait a while to adjust the energy threshold based on the surrounding noise level.

Text to Speech:

Converting Text to Speech using google text to speech api and initialize the language that we need to convert (i.e English). Passing the text and language to the engine, here we have marked slow=False. Which tells the module that the converted audio should have a high speed. Saving the converted audio in a mp3 file.

RESULT DISCUSSION:

If we look into the visualization part in the above section, the scatter plot depicts the individual user id and the number of ratings they have given. In the following graph, it showcases the connection between movie released year and ratings of the movie by user.

Next table is taken from the standard collaborative recommendation evaluation metrics from research paper. But comparing the results with hybrid approach of recommendation system with CV =5 RMSE is 0.8450 and K-Fold is 0.9710 which is pretty good.

RMSE and MAE scores implicates the answer for scalability issue and it can also be improved in future.

INFERENCES:

Collaborative Based Recommendation Technique				
No. of Inputs	Accuracy	Precision	Recall	F-Measure
60	0.79	0.82	0.81	0.8340741
80	0.8	0.81	0.78	0.804031
100	0.81	0.8	0.75	0.7639669
120	0.78	0.77	0.75	0.7439316
140	0.75	0.74	0.68	0.7238938
160	0.76	0.73	0.67	0.7038532
180	0.74	0.7	0.64	0.6737864
200	0.68	0.66	0.6	0.6336842
220	0.68	0.65	0.6	0.6336842
250	0.66	0.65	0.59	0.6136264

Fig. No.: 3 – Collaborative Based Recommendation system[4]

Hybrid Recommendation Technique				
No. of Inputs	Accuracy	Precision	Recall	F-Measure
60	0.92	0.87	0.86	0.8747239
80	0.89	0.86	0.83	0.8447134
100	0.85	0.81	0.83	0.804698
120	0.83	0.8	0.79	0.7846897
140	0.81	0.78	0.75	0.7646809
160	0.79	0.76	0.75	0.7446715
180	0.76	0.71	0.7	0.7146565
200	0.71	0.69	0.67	0.6746341
220	0.71	0.68	0.68	0.6646281
250	0.7	0.68	0.68	0.6546218

Fig. No.: 4 – Hybrid Recommendation Technique[2]

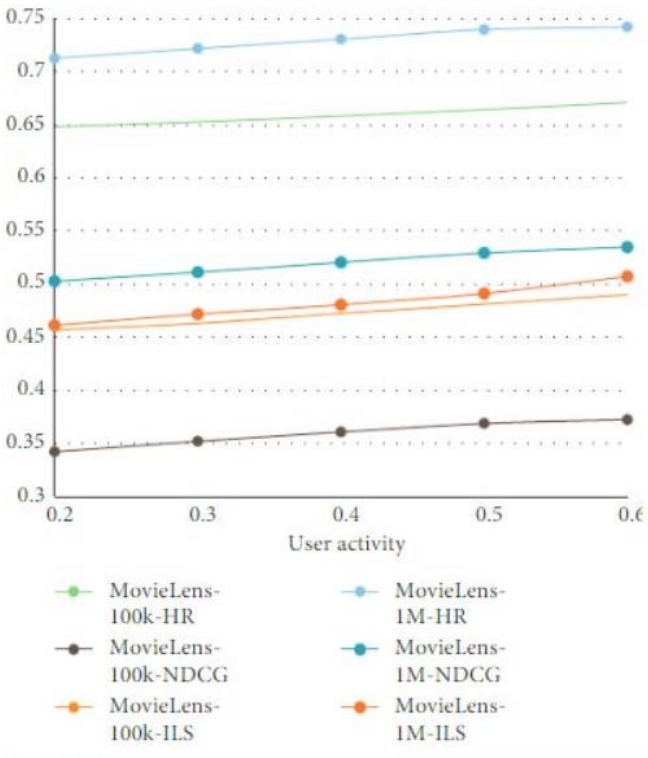


Fig. No.: 5 – User Activity of MovieLens

```
Evaluating RMSE, MAE of algorithm SVD on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    0.9460  0.9579  0.9616  0.9710  0.9499  0.9573  0.0088
MAE (testset)     0.7417  0.7475  0.7438  0.7585  0.7438  0.7470  0.0060
Fit time          0.09    0.09    0.09    0.09    0.09    0.09    0.00
Test time         0.02    0.02    0.02    0.02    0.02    0.02    0.00
{'test_rmse': array([0.94602069, 0.95791714, 0.96160101, 0.97097552, 0.94992721]),
 'test_mae': array([0.74169435, 0.74753881, 0.74376882, 0.75847142, 0.74377132]),
 'fit_time': (0.09404325485229492,
  0.08947324752807617,
  0.09046149253845215,
  0.08910942077636719,
  0.08709311485290527),
 'test_time': (0.015203714370727539,
  0.015218019485473633,
  0.017832517623901367,
  0.018942594528198242,
  0.015253543853759766)}
```

Fig. No.: 6 – RMSE of SVD Algorithm

Load Reader Library and ratings dataset with dataset library. Use SVD algorithm to compute the RMSE algorithm.

It seems to be usual approach but there is a lot of customization algorithms implemented. Another interesting element of the approach is recommendation via speech. It is not implemented in any of the recommendation system. On comparing our algorithmic approach with existing one, our system yielded RMSE score 0.9710 whereas when you refer to Figure no.3 its accuracy score did not exceed 0.81.

CONCLUSION:

Recommender systems open new opportunities of retrieving personalized information on the Internet. It also helps to alleviate the problem of information overload which is a very common phenomenon with information retrieval systems and enables users to have access to products and services which are not readily available to users on the system. We come up with a strategy that focuses on dealing with user's personal interests and based on his previous reviews, movies are recommended to users. This strategy helps in improving accuracy of the recommendations. A personal profile can be created for each user, where each user has access to his own history, his likes, ratings, comments, password modification processes. It also helps in collecting authentic data with improved accuracy and makes the system more responsive.

REFERENCES:

[1]. Joseph A. Konstan, John Riedl. Recommender systems:from algorithms to user experience. User Model User-AdaptInteract, March 2012

[2]. Chenguang Pan, Wenxin Li. Research Paper Recommendation with Topic Analysis. In Computer Design and Applications IEEE, 2010

[3]. Pu P, Chen L, Hu R. A User-Centric Evaluation Framework for Recommender Systems. In: Proceedings of the fifth ACM conference on Recommender Systems, ACM, 2011

[4]. Bhavik Pathak , Robert Garfinkel , Ram D. Gopal , Rajkumar Venkatesan & Fang Yin .Empirical Analysis of the Impact of Recommender Systems on Sales, Journal of Management Information Systems, December 2014

[5]. Al Mamunur Rashid, Istvan Albert, Dan Cosley, Shyong K. Lam, Sean M. McNee, Joseph A. Konstan, John Riedl. Getting to Know You: Learning New User Preferences in Recommender Systems. In: Proceedings of the international conference on intelligent user interfaces, 2012

[6]. J. Ben Schafer, Joseph Konstan, John Riedl. Recommender Systems in E-Commerce. In: Proceedings of the 1st ACM conference on electronic commerce, 2014.

[7]. A. Merve Acilar, Ahmet Arslan. A collaborative filtering method based on artificial immune network. Expert Systems with Application, 2019

[8]. Chen LS, Hsu FH, Chen MC, Hsu YC. Developing recommender systems with the consideration of product profitability for sellers. Int J Inform Sci 2018.

[9]. Jalali M, Mustapha N, Sulaiman M, Mamay A. WebPUM: A Web-based recommendation system to predict user future movements.Exp Syst Applicat, March 2015.

[10]. Fadhel Aljunid, Mohammed & D H, Manjaiah. (2018). Movie Recommender System Based on Collaborative Filter

[11]. https://scikit-learn.org/

[12]. https://scipy.org/

[13].https://www.thepythoncode.com/article/using-speech-recognition-to-convert-speech-to-text-python

[14]. https://pypi.org/project/SpeechRecognition/