CAPSTONE PROJECT WORK REPORT

Phase I

Movie Recommendation System with Machine Learning

RAGU RAMAN S

A report submitted in part fulfilment of the degree of

B.Sc. in Computer Science with Data Analytics

Supervisor: **Mr. R. Anjit Raja**, MCA.,M.Phil.,MCL.,(Ph.D.)
Head of the Department, Dept. of CS with DA

Department of Computer Science with Data Analytics

KPR College of Arts Science and Research
(Affiliated to Bharathiar University, Coimbatore)
Avinashi Road, Arasur, Coimbatore – 641 407

May 2022

CAPSTONE PROJECT WORK REPORT

Phase I

Movie Recommendation System with Machine Learning

Bonafide Work Done by

RAGU RAMAN S

REG. NO. 2028B0035

Dissertation submitted in partial fulfillment of the requirements for the award of Bharathiar University, Coimbatore-46.

Signature of the Guide **Signature of the HOD**

[Mr. R. ANJIT RAJA]

Submitted for the Viva-Voce Examination held on _____

Internal Examiner **External Examiner**

ACKNOWLEDGEMENT

In the accomplishment of completion of my Capstone Project Work Phase- I on Birth Rate Analysis Using Python I would like to convey my special gratitude to Dr. S. Balusamy, Principal of KPR College of Arts Science and Research and Mr. R. Anjit Raja, Assistant Professor and Head, of Department of Computer Science with Data Analytics. Your valuable guidance and suggestions helped me in phase - I of the completion of this project. I will always be thankful to you in this regard. I am ensuring that this project was and not copied.

Student Signature

Place :

Date:

KPR COLLEGE OF ARTS SCIENCE AND RESEARCH

(Affiliated to Bharathiar University, Coimbatore)

Avinashi Road,Arasur,Coimbatore-641407

ABOUT THE COLLEGE

KPR College of Arts Science and Research is the latest addition to the KPR fleet. The College is located in a picturesque campus of about 11. Acres. The College is run by KPR charities under the leadership of our Chairman Dr. K.P. Ramasamy. The KPR Group is one of the largest industrial conglomerate in the country with interest in Textiles, Sugar, Wind Turbines, Automobiles and Education. The College was established in the year 2019 with a vision of providing top class education and life skills to students and thereby serve the nation and beyond. KPRCAS today offers 12 UG programmes in Management, Commerce and Computer Science streams. The Students of KPRCAS undergo intense training not only in the syllabus and curriculum of the affiliating University but are also trained in various areas. So that they emerge as industry ready graduates to meet the varying demands of the competing industries. Character building and Leadership qualities are inculcated into the students to make them responsible citizens focusing on the development of society and nation. A plethora of Clubs and Events encouraged the students to take part in sports and other cultural activities. KPRCAS offers three years undergraduate courses, which are exclusively for Business, Commerce and Computer Science Stream. The students are equipped with skills and knowledge needed to take up various leadership positions and to develop the society. Beyond Book Teaching help them to be professionals. KPRCAS emphasis on making the students academically brilliant, and also prepare them for the real corporate world. The learning curve begins here for the students of KPRCAS.

ABOUT THE DEPARTMENT

Bachelor of Computer Science with Data Analytics (B.Sc. (CS with DA)) was established in the year 2020. Data Analytics helps to raise the quality of data in the entire business system. The goal of data analytics is to construct the means for extracting businessfocused insights from data This requires an understanding of how value and information flows in a business, and the ability to use that understanding to identify business opportunities. The primary aim of a data analyst is to increase efficiency and improve performance by discovering patterns in data. Data analysts exist at the intersection of information technology, statistics and business. They combine these fields in order to help businesses and organizations succeed. The students get exposed to Big Data, Business Intelligence, Data Mining, Data Visualization, Advanced Excel, Predictive Analytics and R Programming.

SYNOPSIS

Recommendation systems are becoming increasingly important in today's extremely busy world. People are always short on time with the myriad tasks they need to accomplish in the limited 24 hours. Therefore, the recommendation systems are important as they help them make the right choices, without having to expend their cognitive resources.

The purpose of a recommendation system basically is to search for content that would be interesting to an individual. Moreover, it involves a number of factors to create personalised lists of useful and interesting content specific to each user/individual. Recommendation systems are Artificial Intelligence based algorithms that skim through all possible options and create a customized list of items that are interesting and relevant to an individual. These results are based on their profile, search/browsing history, what other people with similar traits/demographics are watching, and how likely are you to watch those movies. This is achieved through predictive modeling and heuristics with the data available.

1.INTRODUCTION:

1.1 Recommendation systems are computer programs that suggest recommendations to users depending on a variety of criteria.

These systems estimate the most likely product that consumers will buy and that they will be interested in. Netflix, Amazon, and other companies use recommender systems to help their users find the right product or movie for them.

There are 3 types of recommendation systems.

Demographic Filtering:

The recommendations are the same for every user. They are generalized, not personalized. These types of systems are behind sections like "Top Trending".

Content-based Filtering:

These suggest recommendations based on the item metadata (movie, product, song, etc). Here, the main idea is if a user likes an item, then the user will also like items similar to it.

1. **Collaboration-based Filtering:** These systems make recommendations by grouping the users with similar interests. For this system, metadata of the item is not required.

In the era of information overload, it is very difficult for users to get information that they are really interested in. And for the content provider, it is also very hard for them to make their content stand out from the crowd. That is why many researchers and companies develop Recommender System to solve the contradiction. The mission of Recommender System is to connect users and information, which in one way helps users to find information valuable to them and in another way push the information to specific users. This is the win-win situation for both customers and content providers. VionLabs is a media-tech startup company. The company provides a new way on how consumers are given access to good and suitable content. The mission of VionLabs is to increase needs of its digital user base. Vionel is the movie website developed by VionLabs, which is a place for people who love movies can gather all the information about films in one place[5]. This thesis report will present a more practical recommendation method that can be used on a movie website that does not have enough users.

1.2 Problem Statement

For building a recommender system from scratch, we face several different problems. Currently there are a lot of recommender systems based on the user information, so what should we do if the website has not gotten enough users. After that, we will solve the representation of a movie, which is how a system can understand a movie. That is the precondition for comparing similarity between two movies. Movie features such as genre, actor and director is a way that can categorize movies. But for each feature of the movie, there should be different weight for them and each of them plays a different role for recommendation. So we get these questions:

- How to recommend movies when there are no user information.
- What kind of movie features can be used for the recommender system.
- How to calculate the similarity between two movies.
- Is it possible to set weight for each feature.

1.3 Goals

The goals of this thesis project is to do the research of Recommender Systems and find a suitable way to implement it for Vionel.com. There are many kinds of Recommender Systems but not all of them are suitable for one specific problem and

situation. Our goal is to find a new way to improve the classification of movies, which is the requirement of improving content-based recommender systems.

2 System specification

2.1 hardware configuration

Operating System	Self-Hosted Technical Requirement	Cloud Technical Requirement
Windows	Windows 8.1+	Windows 8.1+
Mac	Mac OS 10.14+	Mac OS 10.14+
Linux	Ubuntu LTS releases 18.04 or later	Ubuntu LTS releases 18.04 or later
RAM	8 GB	
HDD	1 TB	
Processor	64-bit, four-core, 2.5 GHz minimum per core (If your dataset size is significantly larger than the medium dataset, we recommend 8 cores.)	
Mouse	Dell MS116 1000DPI USB Wired Optical Mouse	
Keyboard	Dell KB522 Business Keyboard-Black	
Monitor	Dell 24 Monitor-S2421HN in-Plane Switching (IPS)	

2.2. Software Configuration

IDE	Anaconda
Language Support	Python 3.9
Platform	Jupyter Notebook
Browser	Google Chrome Version 101.0.4951.67
Database	MySQL 8.0.29

3. System Study

3.1 Existing System

What can be recommended?

– Advertising Messages
– Movies
– Books
– Music Tracks
– News Articles
– Restaurants
– Future Friends (Social Network Sites)
– Courses in e-learning
– Jobs
– Research Papers
– Investment Choices
– TV Programs
– Citations
– Clothes

Real-World examples

Here are some of the examples of the pioneers in creating algorithms for recommendation systems and using them to serve their customers better in a personalized manner. These are:

GroupLens:
Helped in developing initial recommender systems by pioneering collaborative filtering model
– It also provided many data-sets to train models including MovieLens and BookLens

Amazon:
Implemented commercial recommender systems
They also implemented a lot of computational improvements

Netflix Prize:
– Pioneered Latent Factor/ Matrix Factorization models

Google-Youtube:
– Hybrid Recommendation Systems
– Deep Learning based systems
– Social Network Recommendations

3.2 Drawbacks

Recommendations are not personalized as per user attributes and all users see the same recommendations irrespective of their preferences
Another problem is that the number of reviews (which reflects the number of people who have viewed the movie) will vary for each movie and hence the average star rating will have discrepancies.
The system doesn't take into account the regional and language preferences and might recommend movies in languages that a regional dialect speaking individual might not understand

A popularity based recommendation system when tweaked as per the needs, audience, and business requirement, it becomes a hybrid recommendation system. Additional logic is added to include customization as per the business needs.

3.3 Proposed System

To build a popularity based recommendation system in Python

Movies.csv has three fields namely:

1. MovieId – It has a unique id for every movie
2. Title – It is the name of the movie
3. Genre – The genre of the movie

The ratings.csv file has four fields namely:

1. Userid – The unique id for every user who has rated one or multiple movies
2. MovieId – The unique id for each movie
3. Rating – The rating given to a user to a movie
4. Timestamp – When was the rating given to a specific movie

```
#import all necessary libraries
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.style.use('seaborn-bright')
%matplotlib inline
#change directory to the folder where data files are present
#This step is not necessary if the data files and jupyter notebook are in same folder
os.chdir(r"C:\Users\mirza\Downloads\Compressed\ml-latest-small\ml-latest-small")
#import ratings file in a pandas dataframe
ratings_data=pd.read_csv("ratings.csv")
ratings_data.head()
```

	userId	movieId	rating	timestamp
0	1	1	4.0	964982703
1	1	3	4.0	964981247
2	1	6	4.0	964982224
3	1	47	5.0	964983815
4	1	50	5.0	964982931

movie_names=pd.read_csv("movies.csv")
movie_names.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

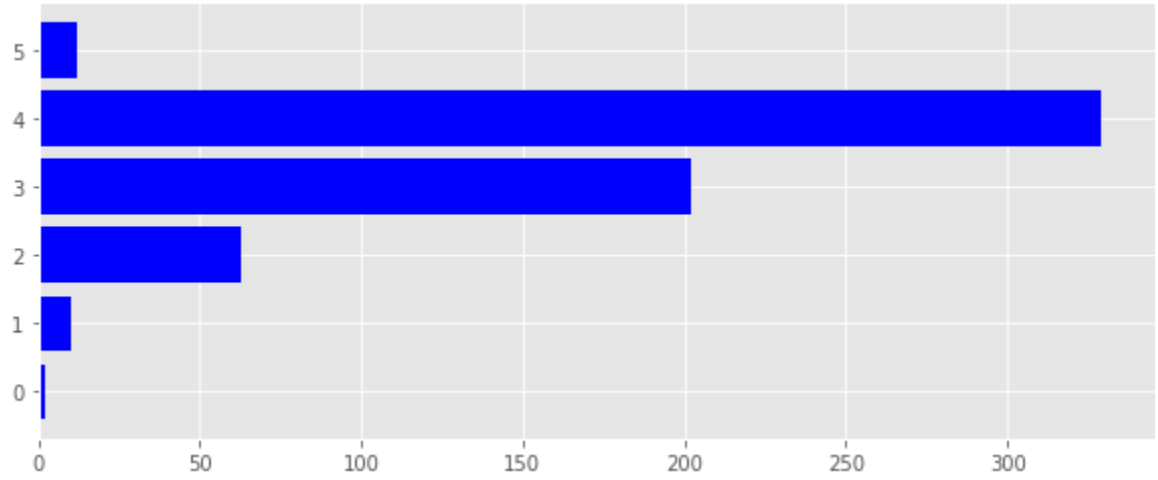
movie_data=pd.merge(ratings_data,movie_names,on='movieId')
movie_data.head()

	userId	movieId	rating	timestamp	title	genres
0	1	1	4.0	964982703	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	5	1	4.0	847434962	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
2	7	1	4.5	1106635946	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
3	15	1	2.5	1510577970	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
4	17	1	4.5	1305696483	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy

#create a dataframe for analysis
trend=pd.DataFrame(movie_data.groupby('title')['rating'].mean())
trend['total number of ratings'] =
pd.DataFrame(movie_data.groupby('title')['rating'].count())
trend.head()

title	rating	total number of ratings
'71 (2014)	4.0	1
'Hellboy': The Seeds of Creation (2004)	4.0	1
'Round Midnight (1986)	3.5	2
'Salem's Lot (2004)	5.0	1
'Til There Was You (1997)	4.0	2

```
#plot rounded-up ratings with number of movies
plt.figure(figsize =(10, 4))
ax=plt.barh(trend['rating'].round(),trend['total number of ratings'],color='b')
plt.show()
```




```
#a bar graph descibibg number of reviews for first 25 movies
plt.figure(figsize =(10, 4))
ax=plt.subplot()
ax.bar(trend.head(25).index,trend['total number of ratings'].head(25),color='b')
ax.set_xticklabels(trend.index,rotation=40,fontsize='12',horizontalalignment="rig
ht")
ax.set_title("Total Number of reviews for each movie")
plt.show()
# Calculate mean rating of all movies and check the popular high rating movies
movie_data.groupby('title')['rating'].mean().sort_values(ascending=False).head()
```

title	
Karlson Returns (1970)	5.0
Winter in Prostokvashino (1984)	5.0
My Love (2006)	5.0
Sorority House Massacre II (1990)	5.0
Winnie the Pooh and the Day of Concern (1972)	5.0
Sorority House Massacre (1986)	5.0
Bill Hicks: Revelations (1993)	5.0
My Man Godfrey (1957)	5.0
Hellbenders (2012)	5.0
In the blue sea, in the white foam. (1984)	5.0
Won't You Be My Neighbor? (2018)	5.0
Red Sorghum (Hong gao liang) (1987)	5.0
Love Exposure (Ai No Mukidashi) (2008)	5.0
My Sassy Girl (Yeopgijeogin geunyeo) (2001)	5.0
The Love Bug (1997)	5.0
Ballad of Narayama, The (Narayama bushiko) (1983)	5.0
Heidi Fleiss: Hollywood Madam (1995)	5.0
Louis Theroux: Law & Disorder (2008)	5.0
Winnie the Pooh Goes Visiting (1971)	5.0
In the Realm of the Senses (Ai no corrida) (1976)	5.0
Winnie Pooh (1969)	5.0
Ex Drummer (2007)	5.0
Tom Segura: Mostly Stories (2016)	5.0
Tom and Jerry: A Nutcracker Tale (2007)	5.0
A Plasticine Crow (1981)	5.0
Tom and Jerry: Shiver Me Whiskers (2006)	5.0
Cosmic Scrat-tastrophe (2015)	5.0
Delirium (2014)	5.0
Lumberjack Man (2015)	5.0
Loving Vincent (2017)	5.0

The primary key here is the movieId which is common in both data files. This key makes it possible to join both these files.
Now, let us have a look at our Python code for popularity based recommendation system.
Step 1: Include the following packages to allow using functions defined under those packages. The cell will include:
– Import os
– Import numpy as np
– Import pandas as pd

Step 2: Change the working directory and replace it with where your dataset is stored

Step 3: Read the ratings file with the below command into the local variable ratings_data. '.head' shows you the top five records in the data set. Also, you can see that we are using the pandas library in this cell which we had called earlier.

Similarly, read the movies file as below

Step 4: Merge the two data variables, ratings_data, and movie_names together by calling merge function from the pandas library on the column movieId. This gives a new data frame 'movie_data'.
Print the movie_data head and you can have a look at the format this new variable appears in.

Step 5: Next, plot a horizontal bar graph using the 'barh' function of the matplotlib library to get an overview of data.We roundup the ratings of all the movies and plot a bar graph of the number of movies against the ratings they got.

Step 6: Next, we plot a bar graph describing the total number of reviews for each movie individually.

Step 7: Finally, we arrange the titles along with their ratings in defending order. This gives us a list of top-rated movies

Now, as mentioned earlier, a large number of users might be reviewing and rating certain movies. While as low as just one user might be rating the other movies. In such cases, some less popular movies can make it to the recommendation list and some of the more popular movies do not make it to the recommendation list. To avoid this bias, one can add a rule to better judge the popularity of a movie. Moreover, newer movies could be more popular than the older ones even though the average ratings might suggest otherwise. In such cases, extra weight.

3.4 Features

Collaborative Filtering

There are two types of collaborative filtering, namely:

1. User – user collaborative filtering
2. Item – item collaborative filtering

Let us understand this type of recommendation system with the help of an example. Say there are two users A and B.
Now, each of these users watched a number of movies and rated them as below:

User A		User B	
Movie	Rating	Movie	Rating
1	–	3	5/5
2	–	4	1/5
3	5/5	6	–
4	1/5	7	–
5	–	8	–

Here, we can see that both A and B have two movies common and both have rated these movies in a similar manner. Hence, one can assume that both these users emanate similar characteristics and would like to see similar movies as each other. Here, the recommendation system will recommend movies 1, 2, and 5 (if rated high) to user B because user A has watched them. Similarly, movies 6, 7, and 8 (if rated high) will be recommended to user A, (if rated high) because user B has watched them. This is an example of user-user collaborative filtering.

Measuring the similarity between users

One can measure the similarity between two users in different ways. A simple way would be to apply Pearson's correlation to the common items. If the result is positively and highly correlated then the movies watched and liked by user A can be

recommended to user B and vice-versa. On the other hand, if the correlation is negative then there is nothing to be recommended as the two users are not alike.

Limitations of user-user collaborative filtering

1. A user might be watching a specific niche type of movies that nobody else is watching. Hence there are no similar profiles resulting in no recommendations.
2. In case of a new movie, there are not enough user ratings to match
3. In the case of a new user, there are not many movies that the user has watched or rated. Hence, it is difficult to map these users to similar users.

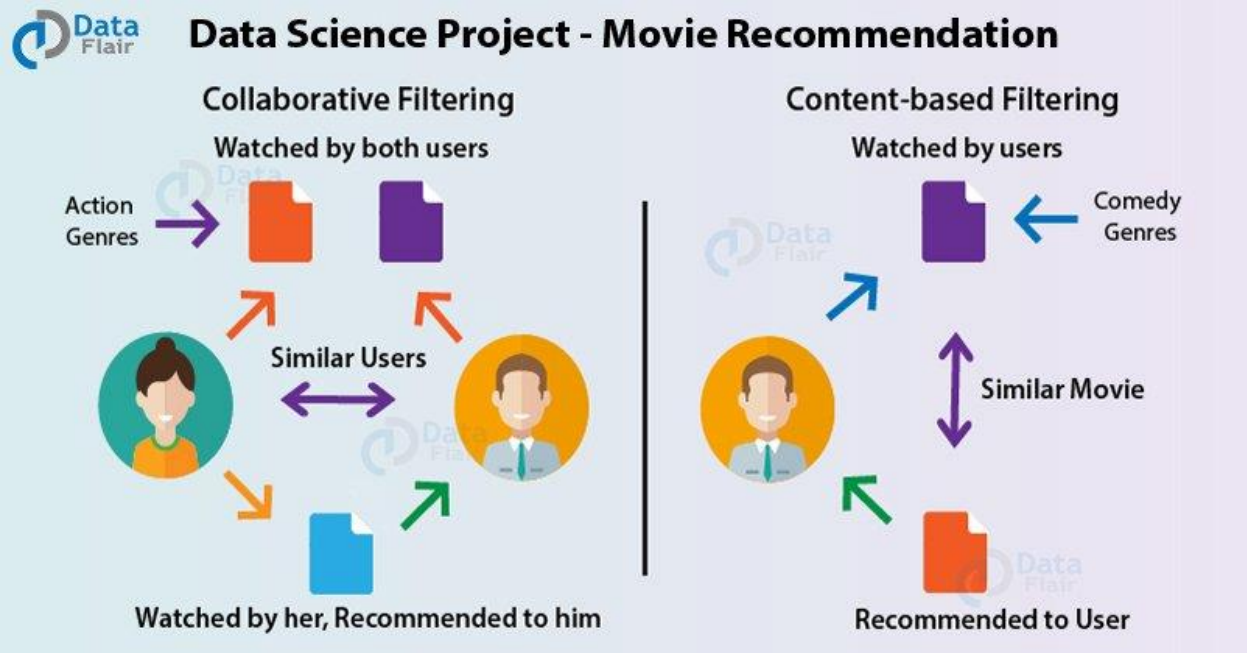
4. System Design

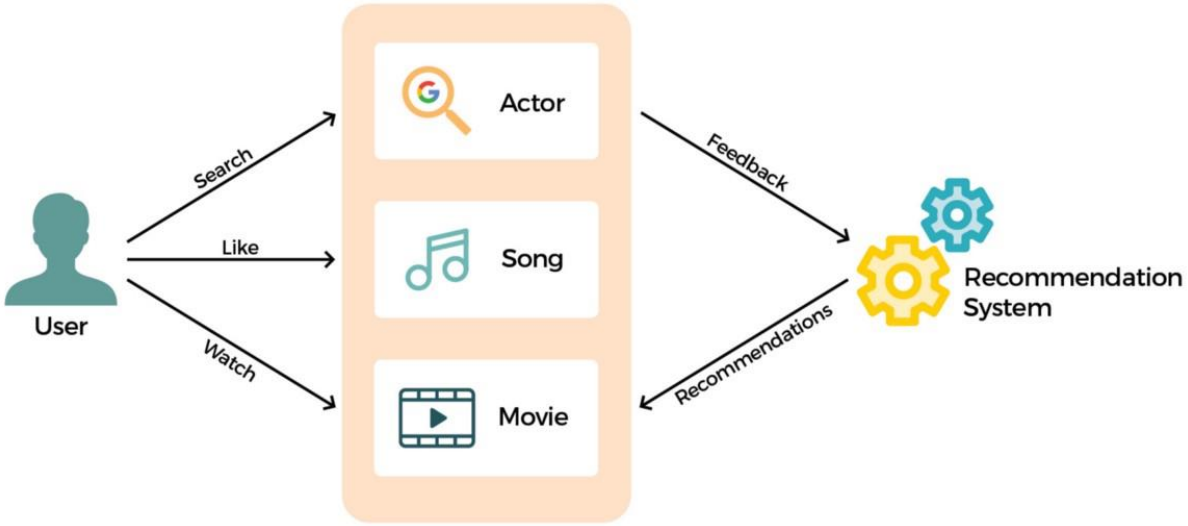
4.1 Form Design

```
#import libraries specific to recommendation system
from surprise import KNNWithMeans
from surprise import Dataset
from surprise import accuracy
from surprise.model_selection import train_test_split
#load the movielens-100k dataset  UserId :: MovieID :: Rating ::Timestamp
data=Dataset.load_builtin('ml-100k')
```

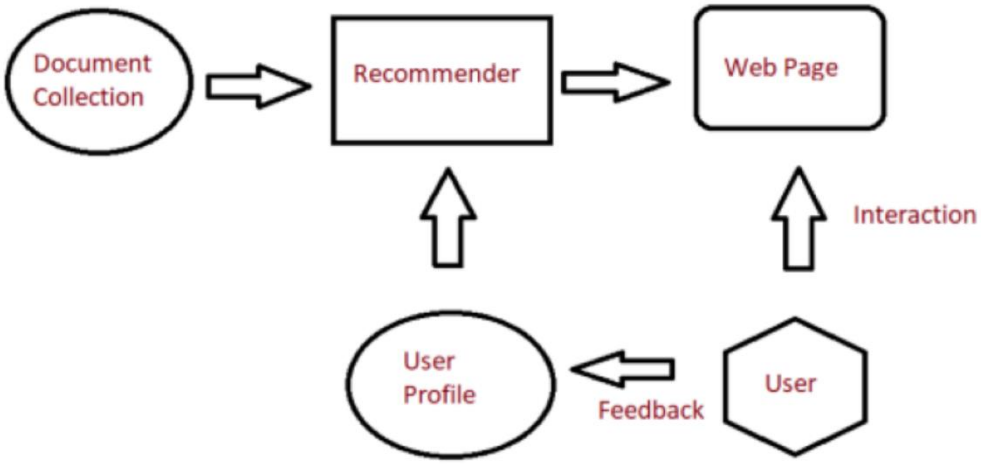
Output:

```
Dataset ml-100k could not be found. Do you want to download it? [Y/n] y
Trying to download dataset from http://files.grouplens.org/datasets/movielens/ml-100k.zip...
Done! Dataset ml-100k has been saved to C:\Users\mirza/.surprise_data/ml-100k
```


Data Flair
Data Science Project - Movie Recommendation
Collaborative Filtering
Watched by both users
Action Genres
Similar Users
Watched by her, Recommended to him
Content-based Filtering
Watched by users
Comedy Genres
Similar Movie
Recommended to User


Actor
Song
Movie
Search
Like
Watch
User
Feedback
Recommendations
Recommendation System

4.2 Input Design



The library function used in order to get user-user collaborative filtering is ‘K nearest neighbours with means. It is a part of a library ‘surprise’, which stands for a simple python library for recommendation systems. ‘Surprise’ also consists of a sub-library called ‘dataset’ which includes some free datasets available to work on. It eliminates the need for downloading datasets from other sources. Another function that is included here is ‘train-test-split’. A portion of the data will be utilized for learning what needs to be recommended and another smaller portion to test the performance of the recommendation system.

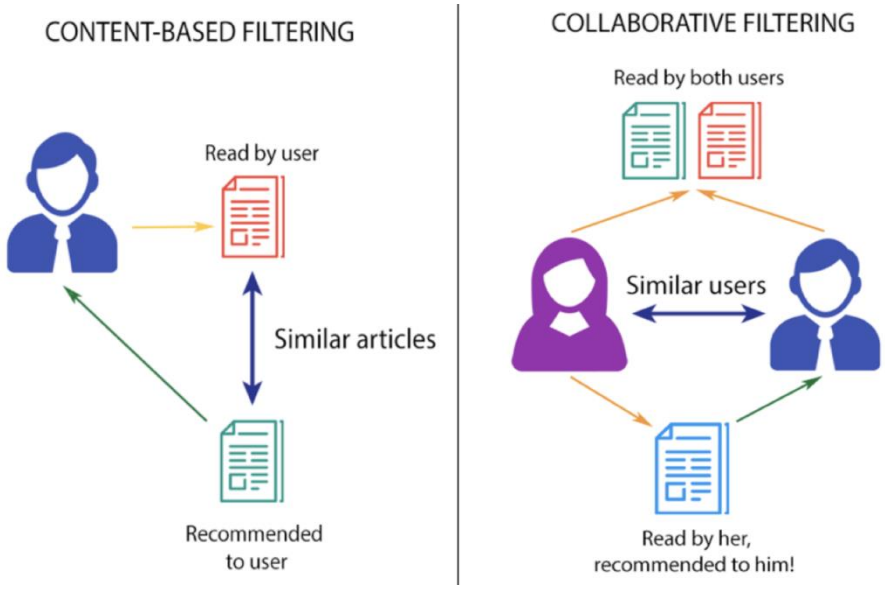
```
#run the trained model against the tesset
test_pred=algo.test(testset)
test_pred
```

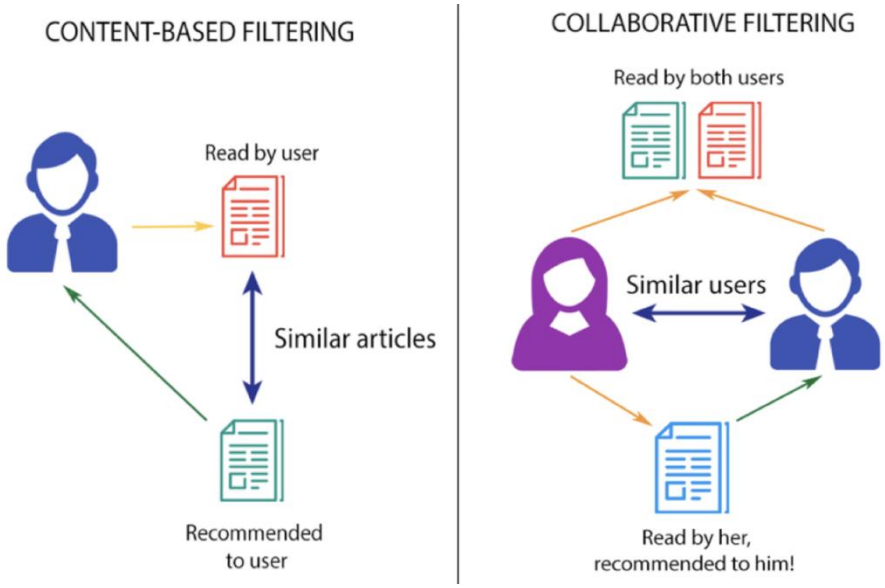
Output:

```
[Prediction(uid='316', iid='306', r_ui=4.0, est=3.7931601072829033, details={'actual_k':
': False}),
 Prediction(uid='130', iid='531', r_ui=5.0, est=4.335278560040126, details={'actual_k': 4
': False}),
 Prediction(uid='899', iid='193', r_ui=3.0, est=3.724571665804737, details={'actual_k': 5
': False}),
 Prediction(uid='181', iid='844', r_ui=1.0, est=1.122843146039015, details={'actual_k': 1
': False}),
 Prediction(uid='479', iid='1028', r_ui=1.0, est=2.7687062213040408, details={'actual_k':
e': False}),
 Prediction(uid='308', iid='1197', r_ui=4.0, est=3.80775134528143, details={'actual_k': 2
': False}),
 Prediction(uid='280', iid='127', r_ui=5.0, est=4.491770725181898, details={'actual_k': 5
': False}),
 Prediction(uid='455', iid='11', r_ui=3.0, est=3.5368786254144218, details={'actual_k': 5
': False}),
 Prediction(uid='92', iid='702', r_ui=3.0, est=3.0942273015464465, details={'actual_k': 2
': False}),
 Prediction(uid='218', iid='42', r_ui=4.0, est=3.468533390067913, details={'actual_k': 50
```

The library function used in order to get user-user collaborative filtering is ‘K nearest neighbours with means. It is a part of a library ‘surprise’, which stands for a simple python library for recommendation systems.

4.3 Output Design





Precision and Recall are two measurements for statistics, which are used to evaluate the quality of statistic result. Precision is used to calculate the ratio of related documents with selected documents. Recall is used to calculate the ratio of related documents with all related documents in selected documents. Below is the definition of the precision and recall under the context of our movie case. Assume T Pi represents the number of test documents belonging to Ci and they are classified to Ci as well. F Pi represents the number of test documents that do not belong to Ci are classified to Ci . F Ni represents the number of test documents belonging to Ci are classified to other categories. So the Precision and Recall in category Ci is defined by: P = T Pi T Pi + F Pi (5.1) R = T Pi T Pi + F Ni (5.2) Generally we should comprehensively consider precision and recall, then we introduce F-Measure, which is defined in Equation 5.3. F = 2 × P × R P + R

id	Unnamed: 0	title	overview	popularity	vote_average	vote_count	release_date	keywords	genres	cast	crew
19404	0	Dilwale Dulhania Le Jayenge	Raj is a rich, carefree, happy-go-lucky second...	31.222	8.7	3323	1995-10-20	[]	['Comedy', 'Drama', 'Romance']	['Shah Rukh Khan', 'Kajol', 'Amrish Puri', 'An...	['Aditya Chopra']
278	1	The Shawshank Redemption	Framed in the 1940s for the double murder of h...	76.654	8.7	20434	1994-09-23	['prison', 'corruption', 'police brutality', '...	['Drama', 'Crime']	['Tim Robbins', 'Morgan Freeman', 'Bob Gunton'...	['Frank Darabont']
238	2	The Godfather	Spanning the years 1945 to 1955, a chronicle o...	75.306	8.7	15270	1972-03-14	['italy', 'loss of loved one', 'love at first ...	['Drama', 'Crime']	['Marlon Brando', 'Al Pacino', 'James Caan', '...	['Francis Ford Coppola']
724089	3	Gabriel's Inferno Part II	Professor Gabriel Emerson finally learns the t...	21.501	8.6	1369	2020-07-31	['based on novel or book']	['Romance']	['Melanie Zanetti', 'Giulio Berruti', 'James A...	['Tosca Musk']
424	4	Schindler's List	The true story of how businessman Oskar Schind...	40.585	8.6	12202	1993-11-30	['based on novel or book', 'factory', 'concent...	['Drama', 'History', 'War']	['Liam Neeson', 'Ben Kingsley', 'Ralph Fiennes...	['Steven Spielberg']

The output after dropping the column:

Changing Data Type

After filling the null values for empty columns, Raghu realizes that he will have to change the data type for most of them:

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 9480 entries, 19404 to 580
Data columns (total 10 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   title         9480 non-null   object 
 1   overview      9464 non-null   object 
 2   popularity    9480 non-null   float64
 3   vote_average  9480 non-null   float64
 4   vote_count    9480 non-null   int64  
 5   release_date  9480 non-null   object 
 6   keywords      9480 non-null   object 
 7   genres        9480 non-null   object 
 8   cast          9480 non-null   object 
 9   crew          9480 non-null   object 
dtypes: float64(2), int64(1), object(7)
memory usage: 814.7+ KB
```

He creates a dictionary with columns as keys and their new type as values. Then, changes the datatype:

```
new_types={'title': str,
 'overview': str,
 'release_date': 'datetime64',}
for col in new_types.keys():
   data[col]=data[col].astype(new_types[col])
```

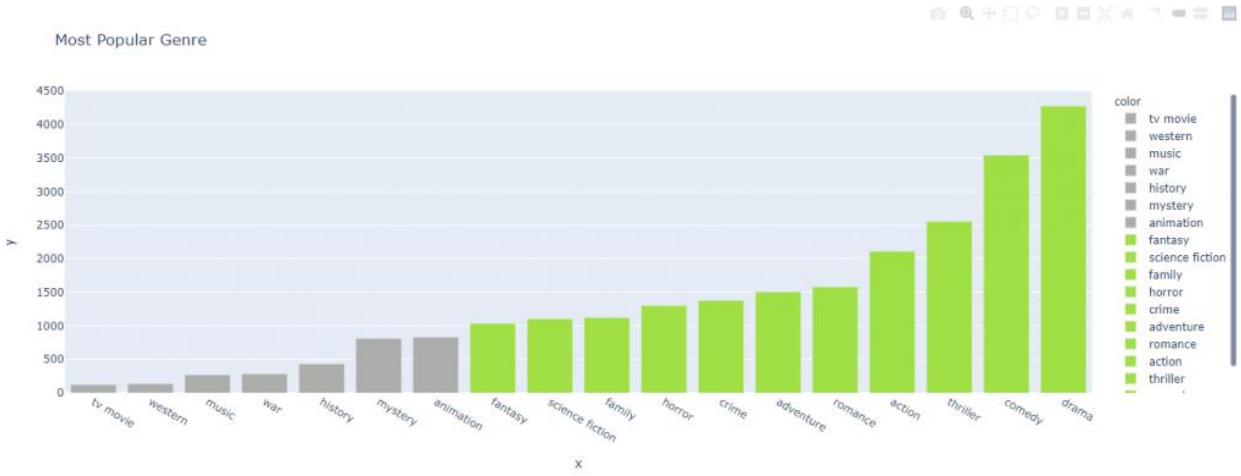
It seems that he has not treated the list columns. The list columns still have some empty values if he changes the type as a **list** directly he will get the following error:

```
~\anaconda3\lib\site-packages\pandas\core\dtypes\missing.py in <genexpr>(.0)
    677         # bytes, generic], Sequence[Union[int, float, complex, str, bytes, generic]],
    678         # Sequence[Sequence[Any]], _SupportsArray]"
--> 679         checker(arr[i : i + chunk_len]).all()  # type: ignore[arg-type]
    680         for i in range(0, total_len, chunk_len)
    681     )

~\anaconda3\lib\site-packages\pandas\core\dtypes\missing.py in <lambda>(x)
    668         # error: Incompatible types in assignment (expression has type "Callable[[Any],
    669         # Any]", variable has type "ufunc")
--> 670         checker = lambda x: _isna_array(  # type: ignore[assignment]
    671             x, inf_as_na=INF_AS_NA
    672         )

~\anaconda3\lib\site-packages\pandas\core\dtypes\missing.py in _isna_array(values, inf_as_na)
    252             result = ~np.isfinite(values)
    253         else:
--> 254             result = np.isnan(values)
    255
    256     return result

TypeError: ufunc 'isnan' not supported for the input types, and the inputs could not be safely coerced to any supported typ
```

The error means that it does not support the **list** datatype as of now. Instead, he creates that column as string type and keeps the values as **comma** separated:

```
for col in ['keywords', 'genres', 'cast', 'crew']:
    for val in ['[',']','\'']:
        data[col]=data[col].str.replace(val,'')
    data[col]=data[col].astype(str)
```

4.4 Database Design



Movie Dataset

The data gathered by Raghu has the following details:

- **Title**: Movie Title.
- **Overview**: Abstract of the Movie.
- **Popularity**: Movie popularity rating as per TMDB.
- **Vote_average**: Votes average out of 10.
- **Vote_count**: Number of votes from the users.
- **Release_date**: Date of release of the movie.
- **Keywords**: Keywords for the movie by TMDB in the list.
- **Genres**: Movie Genres in the list.
- **Cast**: Cast of the movie on the list.
- **Crew**: Crew of the movie in the list.

After cleaning the data, Raghu wants to do some analysis of the data. He creates two functions for list columns:

- **get_unique(data,col):** Returns a list of unique items.

```
def get_uniques(data,col):
    '''
    data: Dataframe object
    col: column name with comma separated values
    ---
    returns: a list of unique category values in that
column
    '''
    out=set([val.strip().lower() for val in
','.join(data[col].unique()).split(',')])
    try:
        out.remove('')
    except:
        return list(out)
    return list(out)
```

- ***get_counts(data,col,categories):*** *Returns the counts for the unique items*

```
def get_counts(data, col, categories):
    '''
    data: dataframe object
    col: name of the column
    categories: categories present
    ----
    return a dictionary with counts of each category
    '''
    categ = {category: None for category in categories}
    for category in tqdm(categories):
        val=0
        for index in data.index:
            if category in data.at[index,col].lower():
                val+=1
        categ[category]=val
    return categ
```

Create a function that takes in the movie title and the cosine similarity score as input and outputs the top 10 movies similar to it.

def get_recommendations(title, cosine_sim=cosine_sim):

```
idx = indices[title]
similarity_scores = list(enumerate(cosine_sim[idx]))
similarity_scores= sorted(similarity_scores, key=lambda x: x[1], reverse=True)
similarity_scores= sim_scores[1:11]
# (a, b) where a is id of movie, b is similarity_scores
movies_indices = [ind[0] for ind in similarity_scores]
movies = movies_df["title"].iloc[movies_indices]
return movies
print("################# Content Based System ##############")
print("Recommendations for The Dark Knight Rises")
print(get_recommendations("The Dark Knight Rises", cosine_sim2))
print()
print("Recommendations for Avengers")
print(get_recommendations("The Avengers", cosine_sim2))
```

Output:

```
############### Content Based System ############
Recommendations for The Dark Knight Rises
65               The Dark Knight
119                Batman Begins
4638   Amidst the Devil's Wings
1196               The Prestige
3073        Romeo Is Bleeding
3326              Black November
1503                      Takers
1986                      Faster
303                    Catwoman
747               Gangster Squad
Name: title, dtype: object

Recommendations for Avengers
7                  Avengers: Age of Ultron
26              Captain America: Civil War
79                               Iron Man 2
169     Captain America: The First Avenger
174                   The Incredible Hulk
85     Captain America: The Winter Soldier
31                               Iron Man 3
33                   X-Men: The Last Stand
68                                 Iron Man
94                 Guardians of the Galaxy
Name: title, dtype: object
```

Conclusion

In this article, we have learned how to create a recommendation system using machine learning. Apart from movie recommendations, you can try making recommender systems from shopping products, news, typing assistance, and so on.

5.00 avg. rating (**95**% score) - **1** vote

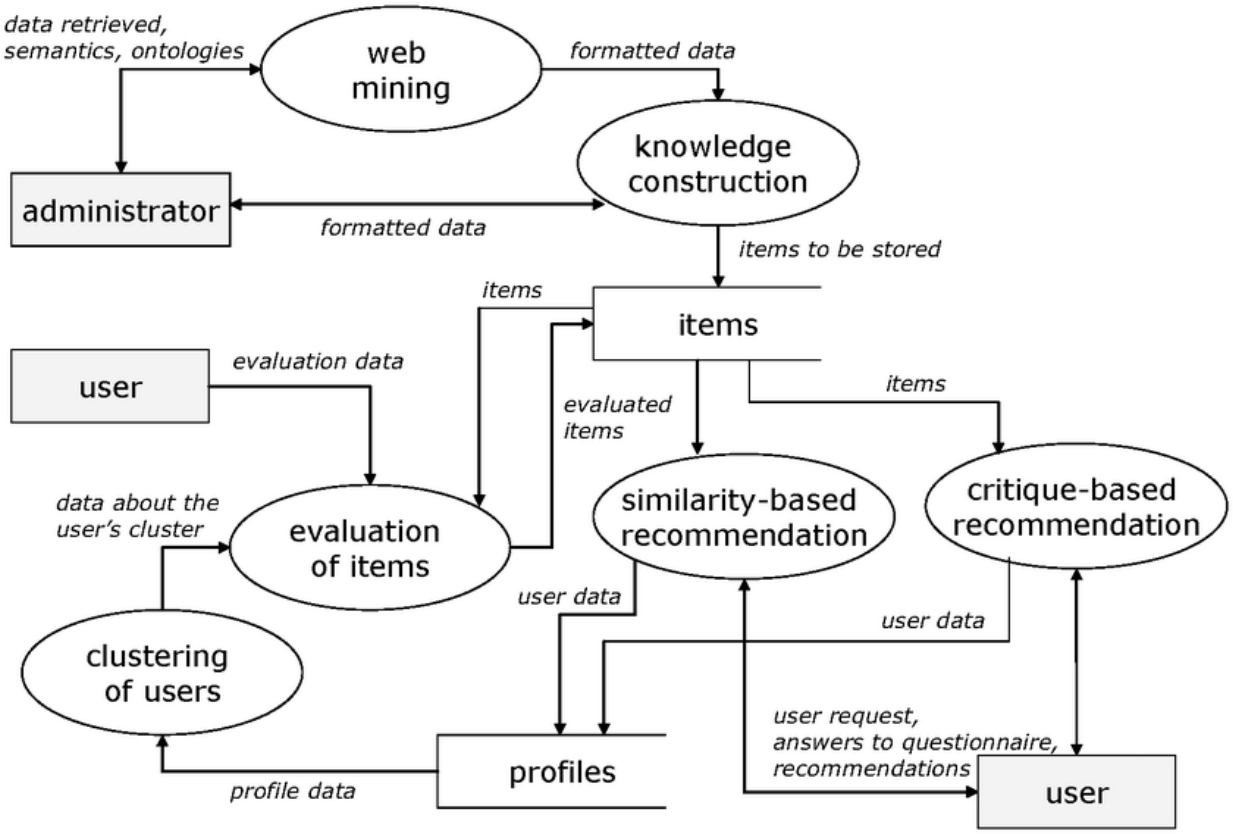
Bibliography

1. https://www.naukri.com/learning/articles/movie-recommendation-system-using-machine-learning/

2. https://www.researchgate.net/figure/Movie-recommender-system-framework_fig4_309896981
3. https://towardsdatascience.com/how-to-build-a-movie-recommendation-system-67e321339109
4. https://data-flair.training/blogs/data-science-r-movie-recommendation/
5. https://www.analyticsvidhya.com/blog/2020/11/create-your-own-movie-movie-recommendation-system/
6. https://www.analyticsvidhya.com/blog/2020/11/create-your-own-movie-movie-recommendation-system/
7. https://labelyourdata.com/articles/movie-recommendation-with-machine-learning
8. https://www.kaggle.com/code/rounakbanik/movie-recommender-systems/notebook
9. https://www.researchgate.net/figure/Movie-recommender-system-framework_fig4_309896981

Appendices

A. Data Flow Diagram



• . Developing A Recommender System On The Basis of Movie Lens Data Set Under the Guidance of: Mr. M VenuGopal Reddy (Associate Prof.)BATCH VIII Sindhu Valavoju (10016T0920) Saigurudatta P.V (10016T0945) Tharun Katanguru (10016T0905) Mounika Parsha (10016T0904) Vikram Konatham (10016T0951)

• 2. ¬Abstract ¬Recommender System ¬Types of Recommender Systems ¬Criteria Followed ¬Hybrid Recommended System ¬Existing System ¬Proposed System ¬Data Flow Diagram ¬Expected Outcome

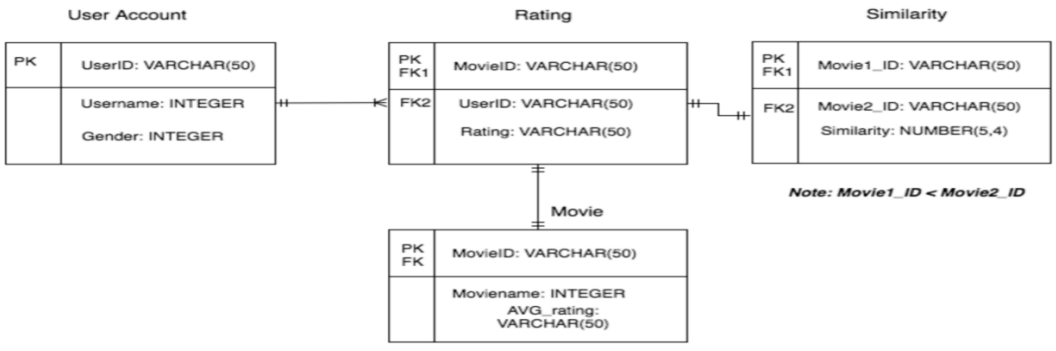
• 3. ABSTRACT: The basic approach that has been proposed to develop a recommender system is COLLOBORATING FILTERING. In CF recommendations for a user are computed based on the k nearest neighbours. A Virtual User is a service

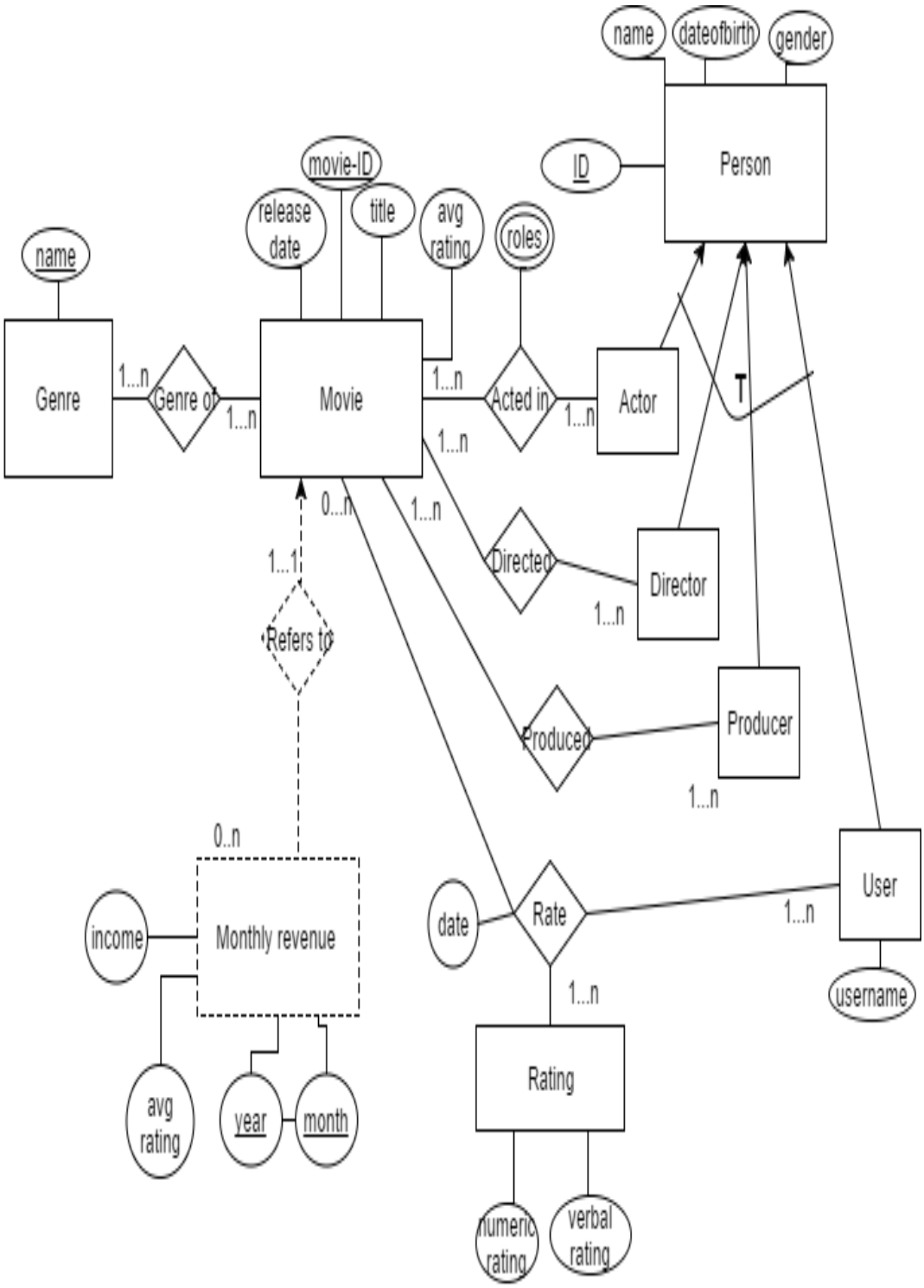
that when executed iteratively, generate traffic from their location to target. Initially, the rating given by a user is divided into categories based on the products which are considered to be given by a VIRTUAL USER. The recommendations of corresponding virtual users of target user are combined for recommendation. This increases the performance of the recommender system and also efficiency in calculating 'k' neighbours.

- 4. Recommender system receives information from the user and recommends the product that fits their needs the best These recommender systems have become a key component of the modern E-Commerce applications. Collaborating Filtering approach has been proposed to build a recommender system. Data set contains three files, movies.dat, ratings.dat and users.dat. Also included are scripts for generating subsets of the data to support rating predictions. Eg: amazon.com uses recommender system to suggest books to the users.

- 5. Algorithm Criteria: 1. Quality of Predictions 2. Speed/Scalability 3. Easily Updated Secondary Criteria: Cold start ability Sparse data handling.

B.Table Structure




name
dateofbirth
gender
Person
ID
movie-ID
release date
title
avg rating
roles
name
Genre
1...n
Genre of
1...n
Movie
1...n
Acted in
1...n
Actor
1...n
0...n
1...n
1...1
Refers to
Directed
1...n
Director
Produced
Producer
1...n
0..n
income
Monthly revenue
avg rating
year
month
date
Rate
1...n
User
1...n
username
Rating
numeric rating
verbal rating

C.SOURCE CODE

```
# Copyright (c) 2020 PaddlePaddle Authors. All Rights Reserved.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
import contextlib
import inspect
import io
import os
import subprocess as sp
import sys
from pathlib import Path
```

```
from setuptools import Command
from setuptools import find_packages
from setuptools import setup
from setuptools.command.develop import develop
from setuptools.command.install import install

HERE = Path(os.path.abspath(os.path.dirname(_file_)))

VERSION = '0.0.0'
COMMITID = 'none'

base = [
    "editdistance",
    "g2p_en",
    "g2pM",
    "h5py",
    "inflect",
    "jieba",
    "jsonlines",
    "kaldiio",
    "librosa==0.8.1",
    "loguru",
    "matplotlib",
    "nara_wpe",
    "onnxruntime",
```

```
    "pandas",
    "paddleaudio",
    "paddlenlp",
    "paddlespeech_feat",
    "praatio==5.0.0",
    "pypinyin",
    "pypinyin-dict",
    "python-dateutil",
    "pyworld",
    "resampy==0.2.2",
    "sacrebleu",
    "scipy",
    "sentencepiece~=0.1.96",
    "soundfile~=0.10",
    "textgrid",
    "timer",
    "tqdm",
    "typeguard",
    "visualdl",
    "webrtcvad",
    "yacs~=0.1.8",
    "prettytable",
    "zhon",
]
```

```
server = [
    "fastapi",
    "uvicorn",
    "pattern_singleton",
    "websockets",
]

requirements = {
    "install":
    base + server,
    "develop": [
        "ConfigArgParse",
        "coverage",
        "gpustat",
        "paddlespeech_ctcdecoders",
        "phkit",
        "Pillow",
        "pybind11",
        "pypi-kenlm",
        "snakeviz",
        "sox",
        "soxbindings",
        "unidecode",
        "yq",
        "pre-commit",
```

```
    ]
}


def check_call(cmd: str, shell=False, executable=None):
    try:
        sp.check_call(
            cmd.split(),
            shell=shell,
            executable="/bin/bash" if shell else executable)
    except sp.CalledProcessError as e:
        print(
            f"{_file_}:{inspect.currentframe().f_lineno}: CMD: {cmd}, Error:",
            e.output,
            file=sys.stderr)
        raise e


def check_output(cmd: str, shell=False):
    try:
        out_bytes = sp.check_output(cmd.split())
    except sp.CalledProcessError as e:
        out_bytes = e.output  # Output generated before error
        code = e.returncode  # Return code
        print(
```

```
        f"{_file_}:{inspect.currentframe().f_lineno}: CMD: {cmd}, Error:",
        out_bytes,
        file=sys.stderr)
    return out_bytes.strip().decode('utf8')


@contextlib.contextmanager
def pushd(new_dir):
    old_dir = os.getcwd()
    os.chdir(new_dir)
    print(new_dir)
    yield
    os.chdir(old_dir)
    print(old_dir)


def read(*names, **kwargs):
    with io.open(
        os.path.join(os.path.dirname(_file_), *names),
        encoding=kwargs.get("encoding", "utf8")) as fp:
       return fp.read()


def _remove(files: str):
    for f in files:
```

```
        f.unlink()


################################### Install
###################################


def _post_install(install_lib_dir):
    # tools/make
    tool_dir = HERE / "tools"
    _remove(tool_dir.glob("*.done"))
    with pushd(tool_dir):
        check_call("make")
    print("tools install.")

    # ctcdecoder
    ctcdecoder_dir = HERE / 'third_party/ctc_decoders'
    with pushd(ctcdecoder_dir):
        check_call("bash -e setup.sh")
    print("ctcdecoder install.")


class DevelopCommand(develop):
    def run(self):
        develop.run(self)
        # must after develop.run, or pkg install by shell will not see
```

```
        self.execute(_post_install, (self.install_lib, ), msg="Post Install...")


class InstallCommand(install):
    def run(self):
        install.run(self)


    # cmd: python setup.py upload
class UploadCommand(Command):
    description = "Build and publish the package."
    user_options = []

    def initialize_options(self):
        pass

    def finalize_options(self):
        pass

    def run(self):
        try:
            print("Removing previous dist/ ...")
            shutil.rmtree(str(HERE / "dist"))
        except OSError:
            pass
```

```
    print("Building source distribution...")
    sp.check_call([sys.executable, "setup.py", "sdist"])
    print("Uploading package to PyPi...")
    sp.check_call(["twine", "upload", "dist/*"])
    sys.exit()


################################### Version
###################################
def write_version_py(filename='paddlespeech/_init_.py'):
    import paddlespeech
    if hasattr(paddlespeech,
               "_version") and paddlespeech.version_ == VERSION:
        return
    with open(filename, "a") as f:
        out_str = f"\n_version_ = '{VERSION}'\n"
        print(out_str)
        f.write(f"\n_version_ = '{VERSION}'\n")

    COMMITID = check_output("git rev-parse HEAD")
    with open(filename, 'a') as f:
        out_str = f"\n_commit_ = '{COMMITID}'\n"
        print(out_str)
        f.write(f"\n_commit_ = '{COMMITID}'\n")

    print(f"{inspect.currentframe().f_code.co_name} done")
```

```
def remove_version_py(filename='paddlespeech/_init_.py'):
    with open(filename, "r") as f:
        lines = f.readlines()
    with open(filename, "w") as f:
        for line in lines:
            if "_version" in line or "commit_" in line:
                continue
            f.write(line)
    print(f"{inspect.currentframe().f_code.co_name} done")


@contextlib.contextmanager
def version_info():
    write_version_py()
    yield
    remove_version_py()


################################### Steup
###################################
setup_info = dict(
    # Metadata
    name='paddlespeech',
    version=VERSION,
```

```
author='PaddlePaddle Speech and Language Team',
author_email='paddlesl@baidu.com',
url='https://github.com/PaddlePaddle/PaddleSpeech',
license='Apache 2.0',
description='Speech tools and models based on Paddlepaddle',
long_description=read("README.md"),
long_description_content_type="text/markdown",
keywords=[
    "speech",
    "asr",
    "tts",
    "speaker verfication",
    "speech classfication",
    "text frontend",
    "MFA",
    "paddlepaddle",
    "beam search",
    "ctcdecoder",
    "deepspeech2",
    "transformer",
    "conformer",
    "fastspeech",
    "vocoder",
    "pwgan",
    "gan",
```

```
    ],
    python_requires='>=3.7',
    install_requires=requirements["install"],
    extras_require={
        'develop':
        requirements["develop"],
        'doc': [
            "sphinx", "sphinx-rtd-theme", "numpydoc", "myst_parser",
            "recommonmark>=0.5.0", "sphinx-markdown-tables", "sphinx-autobuild"
        ],
    },
    cmdclass={
        'develop': DevelopCommand,
        'install': InstallCommand,
        'upload': UploadCommand,
    },

    # Package info
    packages=find_packages(include=('paddlespeech*')),
    zip_safe=True,
    classifiers=[
        'Development Status :: 5 - Production/Stable',
        'Intended Audience :: Developers',
        'Intended Audience :: Science/Research',
        'Topic :: Scientific/Engineering :: Artificial Intelligence',
```

```
        'License :: OSI Approved :: Apache Software License',
        'Programming Language :: Python',
        'Programming Language :: Python :: 3',
        'Programming Language :: Python :: 3.7',
        'Programming Language :: Python :: 3.8',
        'Programming Language :: Python :: 3.9',
    ],
    entry_points={
        'console_scripts': [
            'paddlespeech=paddlespeech.cli.entry:_execute',
            'paddlespeech_server=paddlespeech.server.entry:server_execute',
            'paddlespeech_client=paddlespeech.server.entry:client_execute'
        ]
    })

with version_info():
    setup(**setup_info)
```