OTT RECOMMENDER SYSTEM

Report submitted to the SASTRA Deemed to be University
as the requirement for the course

CSE300 / INT300 MINI PROJECT WORK

Submitted by

RAJALAKSHMI S
(Reg. No.: 122003204)
PURVI
(Reg. No.: 1220150843)

December 2021

School of Computing

SASTRA Deemed to be University
TIRUMALAISAMUDRAM THANJAVUR — 613 401 TAMIL NADU, INDIA

<u>**Bonafide Certificate**</u>

This is to certify that the report titled "**OTT Recommender System**" submitted as a requirement for the course, **CSE300 / INT300 Mini-project Work** for B.Tech. CSE/IT program is a bonafide record of the work done by **Ms. Rajalakshmi S(Reg. No.122003204), Ms. Purvi(Reg.No.122015083)**, during the academic year 2021-22, in the School of Computing, under my supervision.

Signature of Project Supervisor : B Santhi

Name with Affiliation : Dr. Santhi B
Associate Dean - Research
School of Computing
SASTRA Deemed University

Date : 9th January, 2022

Mini-project Viva-voce held on ________________

Examiner 1 **Examiner 2**

SASTRA

ENGINEERING · MANAGEMENT · LAW · SCIENCES · HUMANITIES · EDUCATION

DEEMED TO BE UNIVERSITY

(U/S 3 OF THE UGC ACT, 1956)

THINK MERIT | THINK TRANSPARENCY | THINK SASTRA

SCHOOL OF COMPUTING
THANJAVUR – 613 401

<u>Declaration</u>

We declare that the report titled "**OTT Recommender System"** submitted by me/us is an original work done by us under the guidance of **Dr. Santhi B, Associate Dean - Research, School of Computing, SASTRA Deemed to be University** during the seventh semester of the academic year 2021-22, in the **School of Computing**. The work is original and wherever We have used materials from other sources, we have given due credit and cited them in the text of the report. This report has not formed the basis for the award of any degree, diploma, associate-ship, fellowship, or other similar titles to any candidate of any University.

Signature of the candidate(s) :

Purvi. Rajalakshmi

Purvi **Rajalakshmi S**

Name of the candidate(s) **:** Purvi
Rajalakshmi S

Date **:** 9th January, 2022

ACKNOWLEDGEMENTS

Firstly, we are very grateful for the support and motivation provided to us by every person to put our ideas into a complete picture.

We would also like to express our special thanks to Dr. Santhi B who gave us this opportunity to pursue our project "**OTT Recommender System**" and gave us the necessary guidance and pointers in the time of need.

We would also like to thank our review panel members Dr. Padmakumari and Dr. Brindha V and Dr.Meena for the interest that they have expressed towards our project.

We would also like to thank Dr. Umamakeswari A, Dean of SOC for giving us an opportunity to work with our faculties and get guidance from them and also for making this project a successful one.

We also want to express our greatest gratitude to SASTRA Deemed University for giving us a platform and an opportunity to work on our project which provided us with great knowledge in many areas.

Any project or research works cannot be completed without the support of family and friends and also the god, we are thankful to all.

Thank You.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

1. RS Recommendation System
2. CBF Content-based Filtering
3. CF Collaborative Filtering
4. TF-IDF Term-Frequency Inverse Document Frequency
5. SVD Singular Value Decomposition
6. ML Machine Learning
7. KNN K nearest neighbor
8. RMSE Root Mean Square Error
9. MAE Mean Absolute Error
10. CNN Convolutional neural network

ABSTRACT

Recommendation systems have the potential to change the way websites communicate with users It also allows companies to maximize their ROI (Return on investment) based on the information they can gather on each customer's preferences. A recommendation system is particularly useful for OTT platforms because it boosts the way in which users consume content on the internet.

Motives to make the recommender system :
1. This consumer behavior influence is interesting to work on, and it leads to potential profit.
2. The quest to build a better recommendation system is never ending, and this project is yet another attempt at that.
3. It is one of the most relevant real time applications of data mining and machine learning.

This project implements a few recommendation algorithms (content based,collaborative filtering) An ensemble of these models is then built, to come up with the final recommendation system that predicts accurate movie suggestions for the users.

Other works have implemented collaborative filtering through KNN clustering techniques. However, this project will involve the novel selection of the SVD algorithm implemented using the Surprise Python Scikit tool. It is expected that it will provide a more accurate prediction when compared to KNN.

DATASET
Full Dataset : Consists 26,000,000 ratings, 750,000 tag applications applied to 45,000 movies
Small Dataset: Consists 100,000 ratings and 1,300 tag applications applied to 9,000 movies

CODE
Written in Python. Local command prompt used to run algorithms and web application (built using Flask) displayed on browser

BASE PAPER
Authors : Sudhanshu Kumar , Kanjar De, and Partha Pratim Roy
Journal : IEEE Transactions on computational social systems, 2020

Keywords : Recommender Systems, CF, CBCF, SVD, TF-IDF, hybrid, weighted rating, movie dataset

Guide’s name : Santhi B

CHAPTER 1

Summary of the base paper

This paper presents to us the various ways in which a recommender system can be designed. Recommender systems are used to assist the user in these times of information explosion, mostly from the digital entertainment front including Netflix, Prime Video and Disney+ Hotstar and also from e-commerce websites like Amazon and Flipkart. This paper focuses on building a hybrid recommender system by combining both content-based and collaborative filtering, to recommend users the movies which are liked by users like them and also those movies which are contextually similar to the movie the user likes.

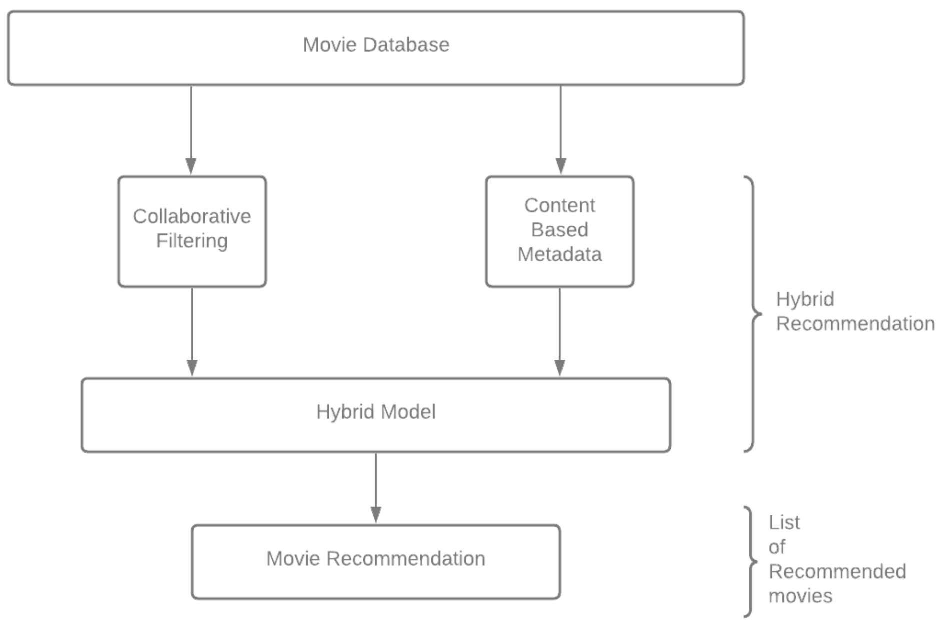


Fig 1.1 Proposed movie recommendation framework

1.1 Proposed System

A) Dataset Description

Experiments were undertaken utilising a variety of public datasets, including the Movielens 100K, Movielens 20M, the Internet Movie Database (IMDb), The Movie Database (TMDB) and the Netflix database. The TMDB dataset was chosen

Attribute	Value
MovieID	0451279
Title	Wonder Woman
Runtime	141 min
Genre	Action,Adventure,Fantasy
Director	Patty Jenkins
Writer	Allan Heinberg
Actors	Gal Gadot',Chris Pine
Rating	7.6 Massachusetts Institute of Technology in 1996.
Production Companies	DC Films,Tencent Pictures
Popularity	524.772
Language	en
Production Countries	United States of America
Budget	816303142

Fig 1.2 Example of a movie entry in the TMDB dataset

B) Content-based and Collaborative Recommendation

A content based recommender system computes similarity between movies based on certain metrics. It suggests movies that are most similar to a particular movie that a user liked. Any content, either in terms of movie description or people involved in the making of the movie is the textual data that is processed to see what other movies have similar content.

Collaborative filtering is a method of making automatic predictions (filtering) about the interests of a user by collecting preferences or taste information from many users (collaborating). SVD, SVD++, K-means clustering are some of the algorithms used to perform matrix factorisation/clustering etc. Ultimately, the goal is to predict a user's preference of a particular movie based on the ratings of other users with tastes similar to him.

C) Hybrid Recommendation

The hybrid recommender combines the features of both content-based similarity and collaborative social filtering. Let f = { f1, f2, ..., fn} and q = {q1, q2, ..., qn} are the content-based feature vectors and weight vectors, respectively. We construct the closeness C of two items i and j as:

$$C(i, j) = \begin{cases} \sum_{n=1}^{N} f_n(A_{n_i}, A_{n_j}), & \text{for } i \neq j \\ 0, & \text{otherwise} \end{cases}$$

where fn (A_{ni}, A_{nj}) corresponds to the similarity between feature values Ani and An j corresponding to two movies.

1.2 Experimental Results and Analysis

Instead of directly forecasting rating levels, the algorithm in many real-world applications makes suitable recommendations. This is known as Top-N recommendation [10], [47], and it recommends certain things to likeable users. For metric evaluation, direct alternative approaches are used (e.g., precision). Precision is measured in terms of movies that the model finds relevant (L_{rel}) and recommends (L_{rec}). Precision@N is defined as follows in the proposed system:

$$\text{Precision@N} = \frac{L_{\text{rel}} \cap L_{\text{rec}}}{L_{\text{rec}}}.$$

CHAPTER 2

Existing techniques used (based on literature review)

Firstly, we analyse the trends/existing techniques used in the field of RS and compare them to the techniques used within the base paper. This gives us insights into the

When it comes to RS, collaborative and content based approaches have always been mostly used. A search system based on document contents and responses collected from other users introduces the concept of CF[1]. There are various problems that need to be handled while developing an RS. Many optimization algorithms, such as gray wolf optimization [3], artificial bee colony [2], and particle swarm optimization [4] have been proposed by researchers. A collaborative movie RS based on gray wolf optimizer and fuzzy c-mean clustering techniques was done by Katarya et al.and Verma [3].The existing framework in [5] was improved by proposing an artificial bee colony and k-mean cluster framework for a collaborative movie RS. This then reduced the scalability and cold start complication. When a hybrid system was further added to it, it showed better accuracy in movie prediction when compared with other movie prediction projects.

2.1 Merits of the base paper

- It uses the hybrid RC algorithm, so it uses the best of CB and CF algorithms.
- The performance metric used is precision, and the scores are very good.
- It uses a huge dataset, which means the model is trained more effectively.
- The dataset is localised, so it is easy for the user to update data and use the algorithm for the data from which he/she wants to make a prediction.
- Experiencing the stark improvement visible in the results upon using a better algorithm-here is no ground truth (it is simple in the sense of being binary) in the results,however this improvement can be realised by the user himself .

2.2 Demerits of the base paper

- The dataset is localised, so any movie out of the scope of the dataset is not recognised.
- The Cold Start problem is discussed in the base paper, but nothing in the implementation works towards resolving the problem.
- The dataset does not contain many languages, so the recommender system is difficult to be used by a diverse user group.
- It does not use other performance metrics like RMSE or MAE.

SOURCE

ott.py

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
from ast import literal_eval
from sklearn.feature_extraction.text import TfidfVectorizer,
CountVectorizer
from sklearn.metrics.pairwise import linear_kernel,
cosine_similarity
from nltk.stem.snowball import SnowballStemmer
from nltk.stem.wordnet import WordNetLemmatizer
from nltk.corpus import wordnet
from surprise import Reader, Dataset, SVD
from surprise.model_selection import cross_validate
from surprise import NormalPredictor
from surprise import KNNBasic
from surprise import Dataset
import tkinter as tk
from tkinter.simpledialog import askstring, askinteger
from tkinter.messagebox import showerror
import time
start_time = time.time()

import warnings; warnings.simplefilter('ignore')


md = pd.read_csv('movies_metadata.csv')

md['genres'] =
md['genres'].fillna('[]').apply(literal_eval).apply(lambda x:
[i['name'] for i in x] if isinstance(x, list) else [])


vote_counts =
md[md['vote_count'].notnull()]['vote_count'].astype('int')
```

```
vote_averages =
md[md['vote_average'].notnull()]['vote_average'].astype('int')
C = vote_averages.mean()


m = vote_counts.quantile(0.95)


md['year'] = pd.to_datetime(md['release_date'],
errors='coerce').apply(lambda x: str(x).split('-')[0] if x != np.nan
else np.nan)


qualified = md[(md['vote_count'] >= m) &
(md['vote_count'].notnull()) &
(md['vote_average'].notnull())][['title', 'year', 'vote_count',
'vote_average', 'popularity', 'genres']]
qualified['vote_count'] = qualified['vote_count'].astype('int')
qualified['vote_average'] =
qualified['vote_average'].astype('int')
qualified.shape



def weighted_rating(x):
    v = x['vote_count']
    R = x['vote_average']
    return (v/(v+m) * R) + (m/(m+v) * C)


qualified['wr'] = qualified.apply(weighted_rating, axis=1)


qualified = qualified.sort_values('wr',
ascending=False).head(250)


s = md.apply(lambda x:
pd.Series(x['genres']),axis=1).stack().reset_index(level=1, drop=True)
s.name = 'genre'
```

```
    gen_md = md.drop('genres', axis=1).join(s)



    def build_chart(movie_name, percentile=0.85):
        genre='Romance'
        for ind in md.index:
            if(md['title'][ind]==movie_name):
                genres=md['genres'][ind]
                genre=genres[0]
                break
        df = gen_md[gen_md['genre'] == genre]
        vote_counts =
df[df['vote_count'].notnull()]['vote_count'].astype('int')
        vote_averages =
df[df['vote_average'].notnull()]['vote_average'].astype('int')
        C = vote_averages.mean()
        m = vote_counts.quantile(percentile)

        qualified = df[(df['vote_count'] >= m) &
(df['vote_count'].notnull()) &
(df['vote_average'].notnull())][['title', 'year', 'vote_count',
'vote_average', 'popularity']]
        qualified['vote_count'] =
qualified['vote_count'].astype('int')
        qualified['vote_average'] =
qualified['vote_average'].astype('int')

        qualified['wr'] = qualified.apply(lambda x:
(x['vote_count']/(x['vote_count']+m) * x['vote_average']) +
(m/(m+x['vote_count']) * C), axis=1)
        qualified = qualified.sort_values('wr',
ascending=False).head(250)

        return qualified.head(10)

    links_small = pd.read_csv('links_small.csv')
    links_small =
links_small[links_small['tmdbId'].notnull()]['tmdbId'].astype('int')
```

```
md = md.drop([19730, 29503, 35587])


md['id'] = md['id'].astype('int')


smd = md[md['id'].isin(links_small)]


smd['tagline'] = smd['tagline'].fillna('')
smd['description'] = smd['overview'] + smd['tagline']
smd['description'] = smd['description'].fillna('')


tf = TfidfVectorizer(analyzer='word',ngram_range=(1, 2),min_df=0,
stop_words='english')
tfidf_matrix = tf.fit_transform(smd['description'])


cosine_sim = linear_kernel(tfidf_matrix, tfidf_matrix)


cosine_sim[0]



smd = smd.reset_index()
titles = smd['title']
indices = pd.Series(smd.index, index=smd['title'])



def get_recommendations(title):
    idx = indices[title]
    sim_scores = list(enumerate(cosine_sim[idx]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1],
reverse=True)
    sim_scores = sim_scores[1:31]
```

```
        movie_indices = [i[0] for i in sim_scores]
        return titles.iloc[movie_indices].head(10)


    credits = pd.read_csv('credits.csv')
    keywords = pd.read_csv('keywords.csv')


    keywords['id'] = keywords['id'].astype('int')
    credits['id'] = credits['id'].astype('int')
    md['id'] = md['id'].astype('int')



    md = md.merge(credits, on='id')
    md = md.merge(keywords, on='id')


    smd = md[md['id'].isin(links_small)]


    smd['cast'] = smd['cast'].apply(literal_eval)
    smd['crew'] = smd['crew'].apply(literal_eval)
    smd['keywords'] = smd['keywords'].apply(literal_eval)
    smd['cast_size'] = smd['cast'].apply(lambda x: len(x))
    smd['crew_size'] = smd['crew'].apply(lambda x: len(x))


    def get_director(x):
        for i in x:
            if i['job'] == 'Director':
                return i['name']
        return np.nan


    smd['director'] = smd['crew'].apply(get_director)


    smd['cast'] = smd['cast'].apply(lambda x: [i['name'] for i in x]
if isinstance(x, list) else [])
```

```
    smd['cast'] = smd['cast'].apply(lambda x: x[:3] if len(x) >=3
else x)


    smd['keywords'] = smd['keywords'].apply(lambda x: [i['name'] for
i in x] if isinstance(x, list) else [])


    smd['cast'] = smd['cast'].apply(lambda x: [str.lower(i.replace("
", "")) for i in x])



    smd['director'] = smd['director'].astype('str').apply(lambda x:
str.lower(x.replace(" ", "")))
    smd['director'] = smd['director'].apply(lambda x: [x])



    s = smd.apply(lambda x:
pd.Series(x['keywords']),axis=1).stack().reset_index(level=1,
drop=True)
    s.name = 'keyword'


    s = s.value_counts()
    s[:5]


    s = s[s > 1]


    stemmer = SnowballStemmer('english')
    stemmer.stem('dogs')



    def filter_keywords(x):
        words = []
        for i in x:
```

```
                if i in s:
                    words.append(i)
            return words



        smd['keywords'] = smd['keywords'].apply(filter_keywords)
        smd['keywords'] = smd['keywords'].apply(lambda x:
[stemmer.stem(i) for i in x])
        smd['keywords'] = smd['keywords'].apply(lambda x:
[str.lower(i.replace(" ", "")) for i in x])




        smd['soup'] = smd['keywords'] + smd['cast'] + smd['director'] +
smd['genres']
        smd['soup'] = smd['soup'].apply(lambda x: ' '.join(x))


        count = CountVectorizer(analyzer='word',ngram_range=(1,
2),min_df=0, stop_words='english')
        count_matrix = count.fit_transform(smd['soup'])


        cosine_sim = cosine_similarity(count_matrix, count_matrix)


        smd = smd.reset_index()
        titles = smd['title']
        indices = pd.Series(smd.index, index=smd['title'])


        def improved_recommendations(title):
            idx = indices[title]
            sim_scores = list(enumerate(cosine_sim[idx]))
            sim_scores = sorted(sim_scores, key=lambda x: x[1],
reverse=True)
            sim_scores = sim_scores[1:26]
            movie_indices = [i[0] for i in sim_scores]
```

```
            movies = smd.iloc[movie_indices][['title', 'vote_count',
'vote_average', 'year']]
            vote_counts =
movies[movies['vote_count'].notnull()]['vote_count'].astype('int')
            vote_averages =
movies[movies['vote_average'].notnull()]['vote_average'].astype('int')
            C = vote_averages.mean()
            m = vote_counts.quantile(0.60)
            qualified = movies[(movies['vote_count'] >= m) &
(movies['vote_count'].notnull()) & (movies['vote_average'].notnull())]
            qualified['vote_count'] =
qualified['vote_count'].astype('int')
            qualified['vote_average'] =
qualified['vote_average'].astype('int')
            qualified['wr'] = qualified.apply(weighted_rating, axis=1)
            qualified = qualified.sort_values('wr',
ascending=False).head(10)
            return qualified.head(10)


      reader = Reader()


      ratings = pd.read_csv('ratings_small.csv')
      ratings.head()



      data = Dataset.load_from_df(ratings[['userId', 'movieId',
'rating']], reader)
      cross_validate(NormalPredictor(), data, cv=5)


      svd = SVD()
      cross_validate(svd, data, measures=['RMSE', 'MAE'], cv=5,
verbose=True)



      trainset = data.build_full_trainset()
```

```
algo = KNNBasic()
algo.fit(trainset)


ratings[ratings['userId'] == 1]



svd.predict(1, 302, 3)


def convert_int(x):
    try:
        return int(x)
    except:
        return np.nan


id_map = pd.read_csv('links_small.csv')[['movieId', 'tmdbId']]
id_map['tmdbId'] = id_map['tmdbId'].apply(convert_int)
id_map.columns = ['movieId', 'id']
id_map = id_map.merge(smd[['title', 'id']],
on='id').set_index('title')


indices_map = id_map.set_index('id')



def hybrid(userId, title):
    idx = indices[title]
    tmdbId = id_map.loc[title]['id']
    movie_id = id_map.loc[title]['movieId']

    sim_scores = list(enumerate(cosine_sim[int(idx)]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1],
reverse=True)
    sim_scores = sim_scores[1:26]
    movie_indices = [i[0] for i in sim_scores]
```

```
            movies = smd.iloc[movie_indices][['title', 'vote_count',
'vote_average', 'year', 'id']]
            movies['est'] = movies['id'].apply(lambda x:
svd.predict(userId, indices_map.loc[x]['movieId']).est)
            movies = movies.sort_values('est', ascending=False)
            return movies.head(10)
```

app.py

```
from flask import Flask, render_template, url_for, request, redirect
from flask_sqlalchemy import SQLAlchemy
from datetime import datetime
from pt import *
from ott import *
import sys

app = Flask(__name__)

@app.route('/', methods=["GET","POST"])

def index():
      return render_template('base.html', data=[{'name':'Simple'},
{'name':'Content Based'}, {'name':'Collaborative'},
{'name':'Hybrid'}])

@app.route("/test" , methods=['GET', 'POST'])
def test():
      algo = str(request.form.get('comp_select'))
      movie = str(request.form.get('movie-name'))
      print(movie, algo)
      sys.stdout = open("example.txt", "w")
      if(algo=='Hybrid'):
            print(hybrid(500, movie))
      if(algo=='Content Based'):
            print(get_recommendations(movie))
      if(algo=='Collaborative'):
            print(improved_recommendations(movie))
      if(algo=='Simple'):
            print(build_chart(movie))
      with open('example.txt', 'r') as f:
```

```
            return render_template('base.html',
data=[{'name':'Simple'}, {'name':'Content Based'},
{'name':'Collaborative'}, {'name':'Hybrid'}], title='OTT',
content=f.read())
      return(str(algo))

if __name__ == "__main__":
      app.run(debug=True)
```

base.html

```
<!DOCTYPE html>
      <head>
            <style>
                  h1 {
                        color: blue;
                        font-family: verdana;
                        font-size: 300%;
                  }
                  pre {
                        color: red;
                        font-size: 100%;
                  }
                  h5 {
                        color: white;
                        padding-left: 20px;
                  }
                  h4 {
                        color: black;

                  }
                  h3 {
                        color: black;
                        padding-left: 15px;
                  }
                  h2 {
                        color: #990000;
                        padding-left: 15px;
                  }
                  p {
                        color: blue;
```

```
                      padding-left: 35px;
                      font-size: 15px;
                 }
           </style>
           <title>OTT</title>
           <meta name="viewport" content="width=device-width,
initial-scale=1.0">
           <link
href="http://netdna.bootstrapcdn.com/bootstrap/3.0.0/css/bootstrap.min
.css" rel="stylesheet" media="screen">
           <style type="text/css">
             .container {
                 max-width: 500px;
                 padding-top: 100px;
             }
           </style>
           <meta name="viewport" content="width=device-width,
initial-scale=1">
             <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"
></script>
             <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.
js"></script>
      </head>
      <body>
      <h2>OTT Recommender System</h2>
      <h3>Description</h3>
      <h4 style="padding:15px">Recommender systems predict content that
a user is more likely to prefer based on previous patterns. Using
recommender systems makes a huge impact on the way users consume
content, hence it is very suitable to use one in OTT platforms. This
project aims to display the significant difference in the quality of
results of the content, collaborative and hybrid filters. It uses RMSE
and MAE as the performance metric for the collaborative filter and
displays the result on a Flask based web application.</h4>
      <nav class="navbar navbar-inverse" role="navigation">
        <div class="container-fluid">
           <div class="navbar-header">
```

```
             <button type="button" class="navbar-toggle"
data-toggle="collapse" data-target="#bs-example-navbar-collapse-1">
                 <span class="sr-only">Toggle navigation</span>
                 <span class="icon-bar"></span>
                 <span class="icon-bar"></span>
                 <span class="icon-bar"></span>
             </button>
           </div>
           <div class="collapse navbar-collapse"
id="bs-example-navbar-collapse-1">
             <form class="form-inline" method="POST" action="{{
url_for('test') }}">
             <div class="form-group">
               <div class="input-group">
                 <h4 style="color:white;">Enter movie name:</h4>
                 <input name="movie-name" id="movie-name" type="text"
class="form-control" placeholder="Search">
                 <h4 style="color:white;">Choose algorithm:</h4>
                       <select name="comp_select" class="selectpicker
form-control">
                         {% for o in data %}
                         <option value="{{ o.name }}">{{ o.name
}}</option>
                         {% endfor %}
                       </select>
               </div>
               <button type="submit" class="btn
btn-default">Go</button>
             </div>
           </form>
           </div>
        </div>
      </nav>
           <pre>{{ content }}</pre>
      </body>
      <h4 style="padding:15px">Project by: <br><br> Purvi (122015083)
<br> Rajalakshmi (122003204) <br> Guide: Santhi B (Associate Dean -
Research)
</h4>
</html>
```

OUTPUT

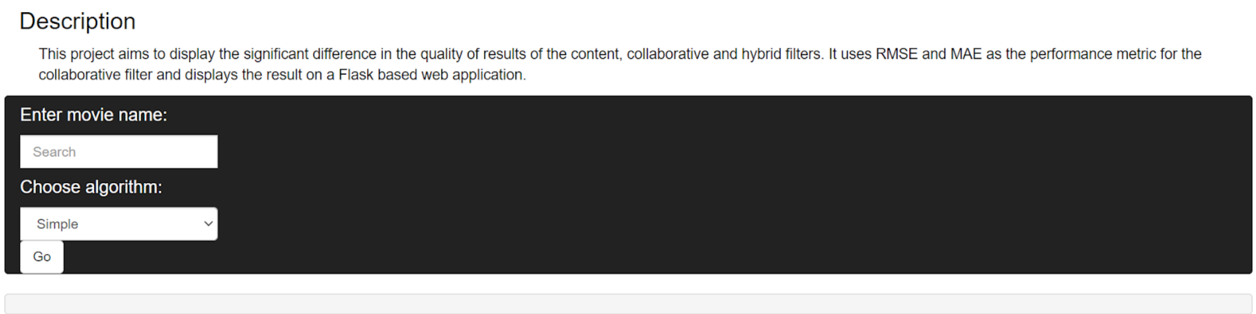
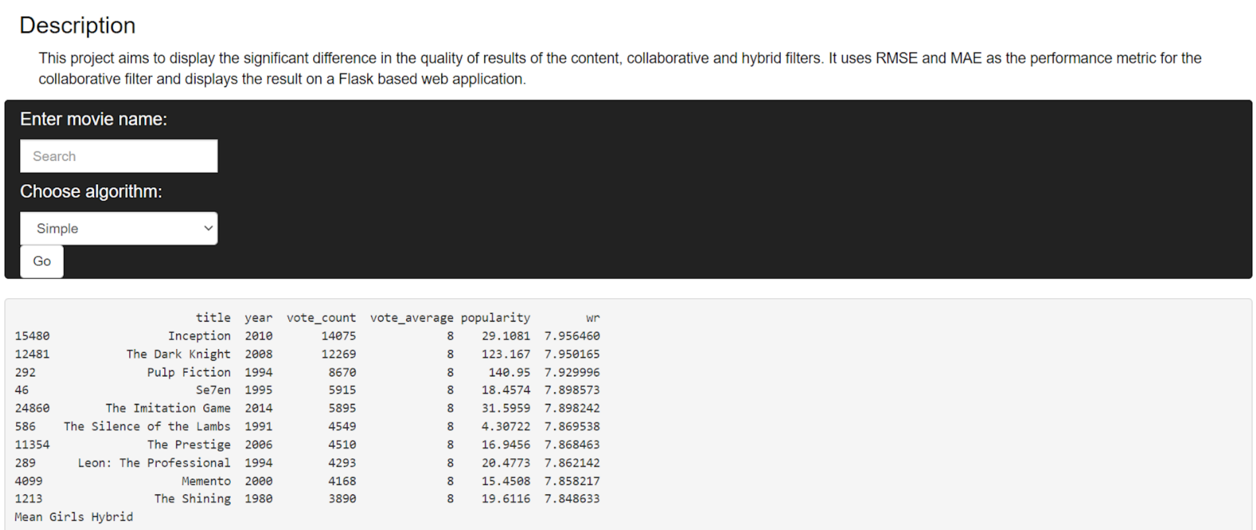


Fig 2.1 Web application view before giving any input



	title	year	vote_count	vote_average	popularity	wr
15480	Inception	2010	14075	8	29.1081	7.956460
12481	The Dark Knight	2008	12269	8	123.167	7.950165
292	Pulp Fiction	1994	8670	8	140.95	7.929996
46	Se7en	1995	5915	8	18.4574	7.898573
24860	The Imitation Game	2014	5895	8	31.5959	7.898242
586	The Silence of the Lambs	1991	4549	8	4.30722	7.869538
11354	The Prestige	2006	4510	8	16.9456	7.868463
289	Leon: The Professional	1994	4293	8	20.4773	7.862142
4099	Memento	2000	4168	8	15.4508	7.858217
1213	The Shining	1980	3890	8	19.6116	7.848633

Mean Girls Hybrid

Fig 2.2 Output that shows a list of recommendations displayed on selecting the algorithm

OTT Recommender System

Description

Recommender systems predict content that a user is more likely to prefer based on previous patterns. Using recommender systems make a huge impact on the way users consume content, hence it is very suitable to use one in OTT platforms. This project aims to display the significant difference in the quality of results of the content, collaborative and hybrid filters. It uses RMSE and MAE as the performance metric for the collaborative filter and displays the result on a Flask based web application.

Enter movie name:

Search

Choose algorithm:

Simple

Go

	title	year	vote_count	vote_average	popularity	wr
15480	Inception	2010	14075	8	29.1081	7.955099
12481	The Dark Knight	2008	12269	8	123.167	7.948610
4863	The Lord of the Rings: The Fellowship of the Ring	2001	8892	8	32.0707	7.929579
7000	The Lord of the Rings: The Return of the King	2003	8226	8	29.3244	7.924031
5814	The Lord of the Rings: The Two Towers	2002	7641	8	29.4235	7.918382
256	Star Wars	1977	6778	8	42.1497	7.908327
1154	The Empire Strikes Back	1980	5998	8	19.471	7.896841
4135	Scarface	1983	3017	8	11.2997	7.802046
9430	Oldboy	2003	2000	8	10.6169	7.711649
1910	Seven Samurai	1954	892	8	15.0178	7.426145

Inception Simple

Project by:
Purvi
Rajalakshmi
Guide: Shanthi B

Fig 2.3 Output of Simple filtering of Inception

OTT Recommender System

Description

Recommender systems predict content that a user is more likely to prefer based on previous patterns. Using recommender systems make a huge impact on the way users consume content, hence it is very suitable to use one in OTT platforms. This project aims to display the significant difference in the quality of results of the content, collaborative and hybrid filters. It uses RMSE and MAE as the performance metric for the collaborative filter and displays the result on a Flask based web application.

Enter movie name:

Search

Choose algorithm:

Simple

Go

	title	vote_count	vote_average	year	wr
4173	Minority Report	2663	7	2002	6.754048
8207	Looper	4777	6	2012	5.937111
7286	X-Men Origins: Wolverine	4086	6	2009	5.927497
7901	Super 8	2496	6	2011	5.888152
7502	The Book of Eli	2207	6	2010	5.875913
6640	Déjà Vu	1519	6	2006	5.832199
5082	Paycheck	594	5	2003	5.103390
7403	Gamer	778	5	2009	5.087694
7828	I Am Number Four	1606	5	2011	5.052101
7903	Green Lantern	2551	5	2011	5.035606

Inception Collaborative

Project by:
Purvi
Rajalakshmi
Guide: Shanthi B

Fig 2.4 Output of Collaborative filtering of Inception

OTT Recommender System

Description

Recommender systems predict content that a user is more likely to prefer based on previous patterns. Using recommender systems make a huge impact on the way users consume content, hence it is very suitable to use one in OTT platforms. This project aims to display the significant difference in the quality of results of the content, collaborative and hybrid filters. It uses RMSE and MAE as the performance metric for the collaborative filter and displays the result on a Flask based web application.

Enter movie name:

Search

Choose algorithm:

Simple

Go

```
5638    Sky Captain and the World of Tomorrow
4596                                 The Core
5580              The Three Lives of Thomasina
7828                           I Am Number Four
4173                           Minority Report
8042                           The Darkest Hour
6640                                   Déjà Vu
6967                                  Doomsday
7948                                Stake Land
477                                 The Shadow
Name: title, dtype: object
Inception Content Based
```

Project by:
Purvi
Rajalakshmi
Guide: Shanthi B

Fig 2.5 Output of Content-based filtering of Inception

OTT Recommender System

Description

Recommender systems predict content that a user is more likely to prefer based on previous patterns. Using recommender systems make a huge impact on the way users consume content, hence it is very suitable to use one in OTT platforms. This project aims to display the significant difference in the quality of results of the content, collaborative and hybrid filters. It uses RMSE and MAE as the performance metric for the collaborative filter and displays the result on a Flask based web application.

Enter movie name:

Search

Choose algorithm:

Simple

Go

```
                  title  vote_count  vote_average  year     id       est
8207             Looper      4777.0           6.6  2012  59967  3.452580
4173    Minority Report      2663.0           7.1  2002    180  3.227893
5296             Zardoz       106.0           5.8  1974   4923  3.206885
7502    The Book of Eli      2207.0           6.6  2010  20504  3.073153
7901            Super 8      2496.0           6.6  2011  37686  3.057755
477          The Shadow       140.0           5.4  1994   8850  3.038709
7828   I Am Number Four      1606.0           5.9  2011  46529  3.030347
7208          Replicant        93.0           5.0  2001  10596  3.029252
7403              Gamer       778.0           5.6  2009  18501  3.003971
5970  Cube²: Hypercube       383.0           5.4  2002    437  2.986986
Inception Hybrid
```

Project by:
Purvi
Rajalakshmi
Guide: Shanthi B

Fig 2.6 Output of Hybrid filtering of Inception

CHAPTER 3

3.1 Conclusions

The Recommender system is a useful tool that helps creators influence the way users consume content on the internet. Analysing user preference patterns and the similarity of content on a dataset can lead to a boost in usage of the application.

In this project, a recommender system was built that makes predictions of what a user would like based on the given input of a movie name. It implemented various algorithms like simple recommender, content-based and collaborative filtering. Finally, a hybrid recommender was built with the CF and CBF algorithms to make a better prediction.

Apart from gaining insights on how the recommender system works on a real dataset, it is learnt how different algorithms shape the prediction made by our RS.

3.2 Future scope and plans

Building the perfect RS is a never ending project. There is always room for improvement, and various Outstanding factors still remain to be addressed even with the best of them. Some areas where the extension of this project will be fruitful are :

- Expanding the dataset such that other languages and a wider range of movies is considered. This will fetch better results.
- Usage of sentimental analysis from user reviews can be an additional feature of the model that will boost the accuracy.
- Better performance metrics can be used, and the GUI can be modified to let the user choose the movie name on a dropdown instead of typing the name.

CHAPTER 4

4.1 References in Chapter 1

[1] P. Cremonesi, Y. Koren, and R. Turrin, "Performance of recommender algorithms on top-N recommendation tasks," in Proc. 4th ACM Conf. Rec. Syst. (RecSys), 2010, pp. 39–46.

[2] A. Said, A. Bellogín, and A. D. Vries, "A top-N recommender system evaluation protocol inspired by deployed systems," in Proc. LSRS Workshop ACM RecSys, 2013, pp. 1–7.

4.2 References in Chapter 2

[1] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, "Using collaborative filtering to weave an information tapestry," Commun. ACM, vol. 35, no. 12, pp. 61–70, Dec. 1992.

[2] C.-C. Hsu, H.-C. Chen, K.-K. Huang, and Y.-M. Huang, "A personalized auxiliary material recommendation system based on learning style on Facebook applying an artificial bee colony algorithm," Comput. Math. Appl., vol. 64, no. 5, pp. 1506–1513, Sep. 2012.

[3] R. Katarya and O. P. Verma, "Recommender system with grey wolf optimizer and FCM," Neural Comput. Appl., vol. 30, no. 5, pp. 1679–1687, Sep. 2018.

[4] S. Ujjin and P. J. Bentley, "Particle swarm optimization recommender system," in Proc. IEEE Swarm Intell. Symp. (SIS), 2003, pp. 124–131.

[5] R. Katarya, "Movie recommender system with Metaheuristic artificial bee," Neural Comput. Appl., vol. 30, no. 6, pp. 1983–1990, Sep. 2018.

Peer Evaluation Form for Group Work

Write the name of each of your group members in a separate column. For each person, indicate the extent to which you agree with the statement on the left, using a scale of 1-4 (1=strongly disagree; 2=disagree; 3=agree; 4=strongly agree). Total the numbers in each column.

Evaluation Criteria	Group member: Purvi	Group member: Rajalakshmi
Attends group meetings regularly and arrives on time.	4	4
Contributes meaningfully to group discussions.	4	4
Completes group assignments on time.	4	4
Prepares work in a quality manner.	4	4
Demonstrates a cooperative and a supportive attitude.	4	4
Contributes significantly to the success of the mini-project.	4	4
TOTALS	24	24

Feedback on team dynamics:

1. How effectively did your group work?

We have worked towards the success of the project by conducting frequent team meetings in google meet to brainstorm ideas and to share what we have learnt. The entire project was done together by us and each of us contributed towards the success of this project.

2. Were the behaviors of any of your team members particularly valuable or detrimental to the team? Explain.

Working with my teammate was very valuable, since our work complemented each other's, and division of work among us was really easy too.

3. What did you learn about working in a group from this mini-project that you will carry into your next group experience?

The most important thing which we learnt is even though in teams we have different ideas we discussed and sorted it out to one specific thing which satisfies both the members and then worked towards it. This helped us in completing the project at the right time without having any internal arguments and also to work under some pressure. We have also gained good knowledge on various technical domains. It taught me that working together makes you gain various perspectives and insights.

Self-Evaluation Form for Group Work

Name- Rajalakshmi

	Seldom	Sometimes	Often
Contributed good Ideas	-	yes	-
Listened to and respected the ideas of others	-	-	yes
Compromised and cooperated	-	-	yes
Took initiative where needed	-	yes	-
Came to meetings prepared	-	-	yes
Communicated effectively with teammates	-	-	yes
Did my share of the Work	-	-	yes

My greatest strengths as a team member are: analyzing code and research work,writing skills

The group work skills I plan to work to improve are: I will try to meet deadlines beforehand in future.

Name - Purvi

	Seldom	Sometimes	Often
Contributed good Ideas	-	-	yes
Listened to and respected the ideas of others	-	-	yes
Compromised and cooperated	-	yes	-
Took initiative where needed	-	-	yes
Came to meetings prepared	-	-	yes
Communicated effectively with teammates	-	-	yes
Did my share of the Work	-	-	yes

My greatest strengths as a team member are: Noting down required things, time management.
The group work skills I plan to work to improve are: Need to discuss more with the teammates regarding the ideas, to submit things on time.

APPENDIX - Base paper

Movie Recommendation System Using Sentiment Analysis From Microblogging Data
Sudhanshu Kumar , Kanjar De, and Partha Pratim Roy IEEE Transactions on computational social systems, 2020