Recommendation system

INDUSTRIAL TRAINING REPORT

SUBMITTED BY

M. RAGUPATHI
REG.NO:191011030
THIRD YEAR

In partial fulfilment for the requirements of the award of the Degree of

BACHELOR OF ENGINEERING
IN
COMPUTER SCIENCE AND ENGINEERING(DATA SCIENCE)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING FACULTY OF ENGINEERING AND TECHNOLOGY ANNAMALAI UNIVERSITY
ANNAMALAI NAGAR – 608 002

NOVEMBER – 2021

ANNAMALAI UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

This is to certify that this Report titled ***"Recommendation system"*** *is a bonafide record of the work done by* ***M. RAGUPATHI*** *(Reg. No:191011030) in partial fulfillment of the requirements for the award of the Degree of* ***Bachelor of Engineering in Computer Science and Engineering (Data Science)*** *during the odd semester of the academic year 2021 to 2022.*

Dr. L. R. SUDHA
Associate Professor
Dept. of Computer Science & Engg.

Dr. S. PALANIVEL
Professor and Head
Dept. of Computer Science & Engg.

Place: Annamalai Nagar

Date:29-10-2021

Internal Examiner

External Examiner

TABLE OF CONTENTS

ACKNOWLEDGMENTS

A great deal of arduous work and efforts has been spent in implementing this project work. Several special people have guided us and have contributed significantly to this work and so this becomes obligatory to record our thanks to them.

We would like to thank our respected **DEAN Dr. A. MURAGAPPAN M.E., Ph.D** for allowing us to do this project and providing require time to complete the same.

I wish to express our sincere gratitude to **Dr. S. PALANIVEL, M.E., Ph.D Head Of Computer Science And Engineering Department** for her extended enourage to fulfill this project.

I express our sincere thanks to **Dr.L.R.SUDHA M.E., Ph.D** Project Coordinator for the useful suggestions, which helped us for completing the project work in time.

I would like to extend our sincere thanks to **Dr.L.R.SUDHA M.E., Ph.D** Supervisor, for giving this opportunity to do this project and also for her inspiring guidance, generous help and support.

I would like to extend our sincere thanks to all our department staff members and to our parents for their advice and encouragement to do the project work with full interest.

I would like to express my greatest appreciation to the all individuals who have helped and supported me throughout the project.

A special acknowledgment goes to my classmates who helped me in completing the project by exchanging interesting ideas and sharing their experience.

ABSTRACT

On the Internet, where the number of choices is overwhelming, there is need to filter, prioritize and efficiently deliver relevant information in order to alleviate the problem of information overload, which has created a potential problem to many Internet users. Recommender systems solve this problem by searching through large volume of dynamically generated information to provide users with personalized content and services. This paper explores the different characteristics and potentials of different prediction techniques in recommendation systems in order to serve as a compass for research and practice in the field of recommendation systems.

In this project, developed a recommendation system that utilizes not only the user's demographics.

Recommendation systems are a way of suggesting like or similar items and ideas to a user. It is an information filtering technique, which provides users with information, which may be interested in.

INTRODUCTION

In this age of information overload, people use a variety of strategies to make choices about what to buy, how to spend their leisure time or what to read. Search engines help us a little bit. But Recommendation systems automate some of these strategies with the goal of providing affordable, personal, and high-quality recommendations.

Recommendation systems are a way of suggesting like or similar items and ideas to a user. It is an information filtering technique, which provides users with information, which may be interested in.

Search engines focused more on Information retrieval but recommendation system focused on Information Filtering. In search engines you see a query box where you type in what you're looking for and they bring back a list of results. But in recommendation system you don't build a query, Recommendations engines observe your actions and construct queries for you so some content has just appeared on your screen that is relevant to you but you didn't request it.

Need of Recommendation systems

Value for the customer

Find things that are interesting:

Recommendation systems provide affordable, personal, and high-quality recommendations, which he/she may be interested in.

Narrow down the set of choices:

A company with an inventory of thousands and thousands of items would be hard to product suggestions for all of its products, Recommendation system will narrow down the choices.

Help me explore the space of options:

People generally like to be recommended things which they would like, and when they use a site which can relate to his/her choices extremely perfectly then he/she is bound to visit that site again.

***Discover new things*:**

Recommendation systems recommend products which are similar to the ones that a user has liked in the past. If you like an item you will also like a ‘similar’ item.

Value for the provider

Unique personalized service for the customer:

Companies using recommendation systems focus on increasing sales as a result of very personalized offers and an enhanced customer experience.

***Increase trust and customer loyalty*:**

The user starts to feel known and understood and is more likely to buy additional products or consume more content which increase customer loyalty.

Increase sales, click trough rates, conversion:

Recommendations typically speed up searches and make it easier for users to access content they're interested in, and surprise them with offers they would have never searched for.

***Opportunities for promotion, persuasion*:**

Companies are able to gain and retain customers by sending out emails with links to new offers that meet the recipients' interests, or suggestions of films and TV shows that suit their profiles.

Obtain more knowledge about customer:

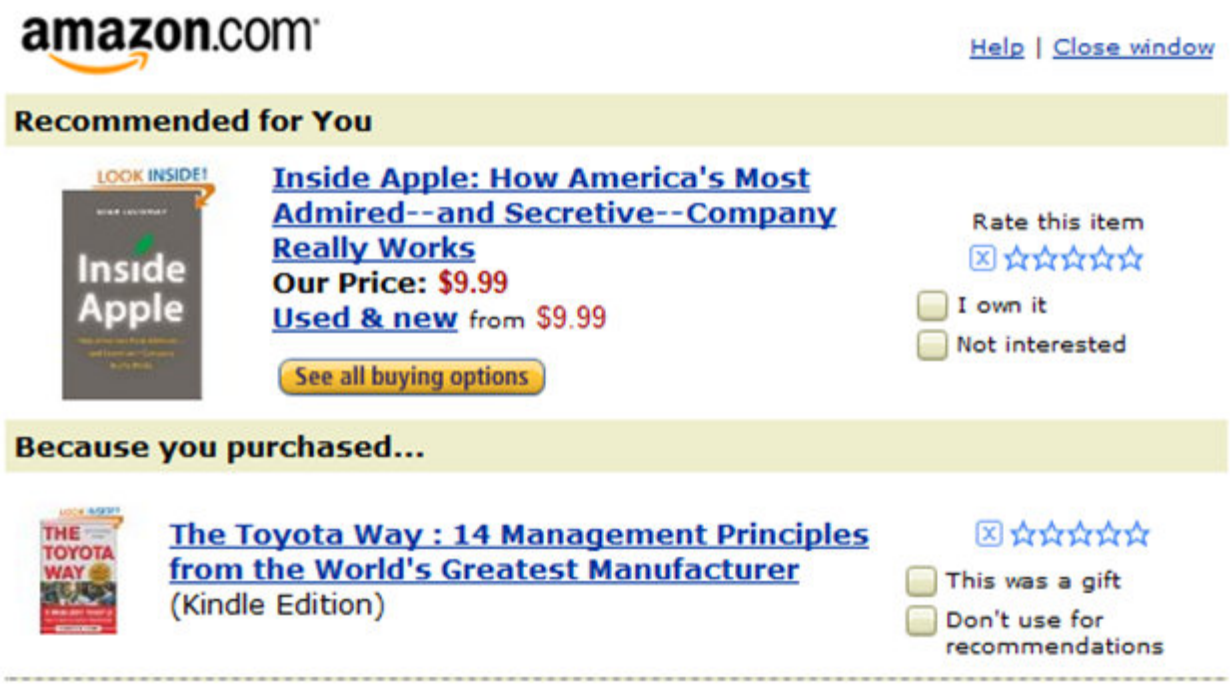
By knowing what a user wants, the company gains competitive advantage and the threat of losing a customer to a competitor decrease

Area of use

An increasing number of online companies are utilizing recommendation systems to increase user interaction and enrich shopping potential. Use cases of recommendation systems have been expanding rapidly across many aspects of eCommerce and online media over the last 4-5 years.Current recommendation engine use-cases at Amazon, Netflix, YouTube, and others

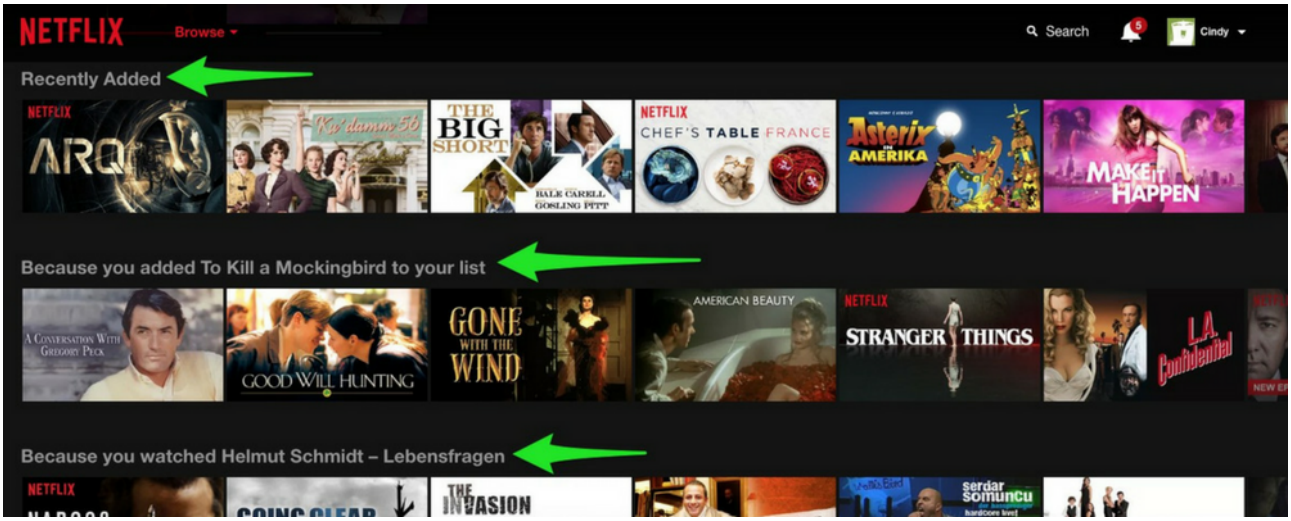
Amazon

At Amazon use recommendation algorithms to personalize the online store for each customer. The store radically changes based on customer interests. When a customer clicks on the "your recommendations" the link leads to another page where recommendations may be filtered even further by subject area, product types, and ratings of previous products and purchases. According to McKinsey & Company, **35 percent** of Amazon's revenue is generated by its recommendation engine.



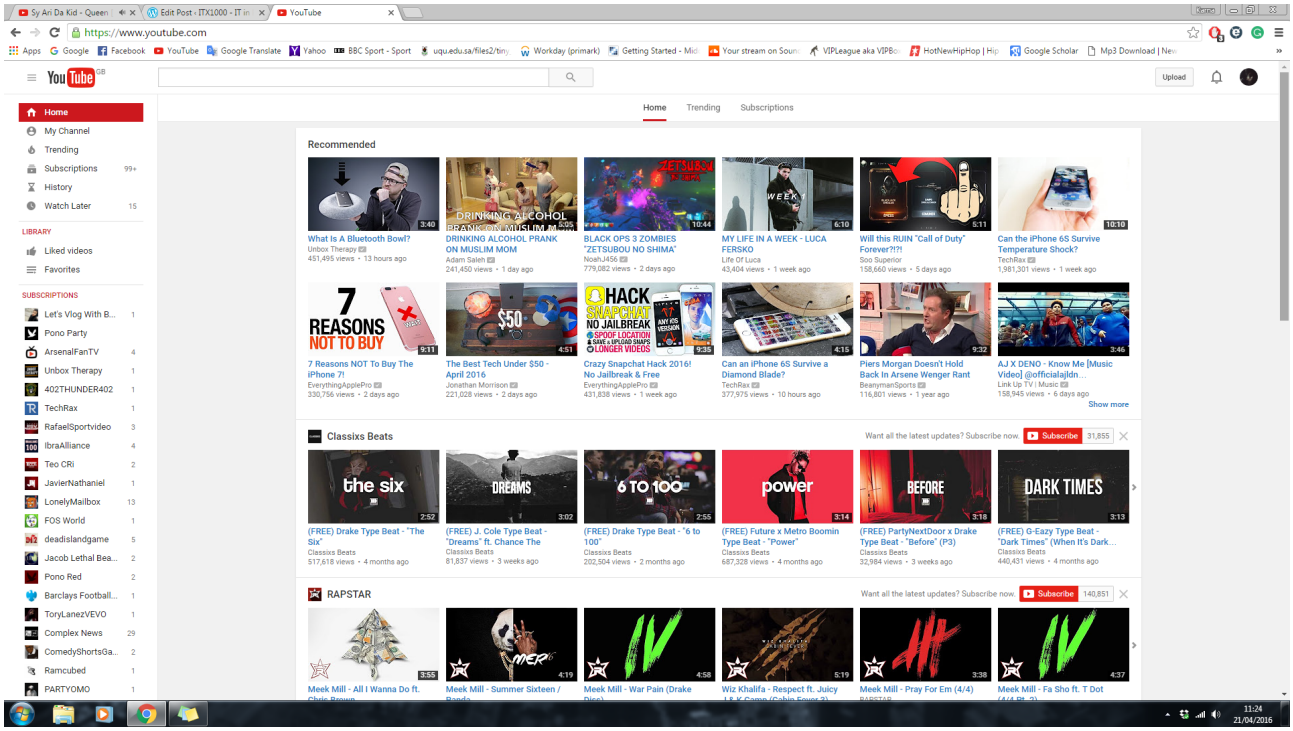
Netflix

Netflix uses Recommendation systems personalized diversity to generate Top Ten recommendations for user households, so that it can offer videos that each member of the household may be interested in. The company also focuses on awareness and promoting trust to help develop its personalized approach. According to McKinsey, **80 percent** of what users watch on Netflix come from product recommendations.

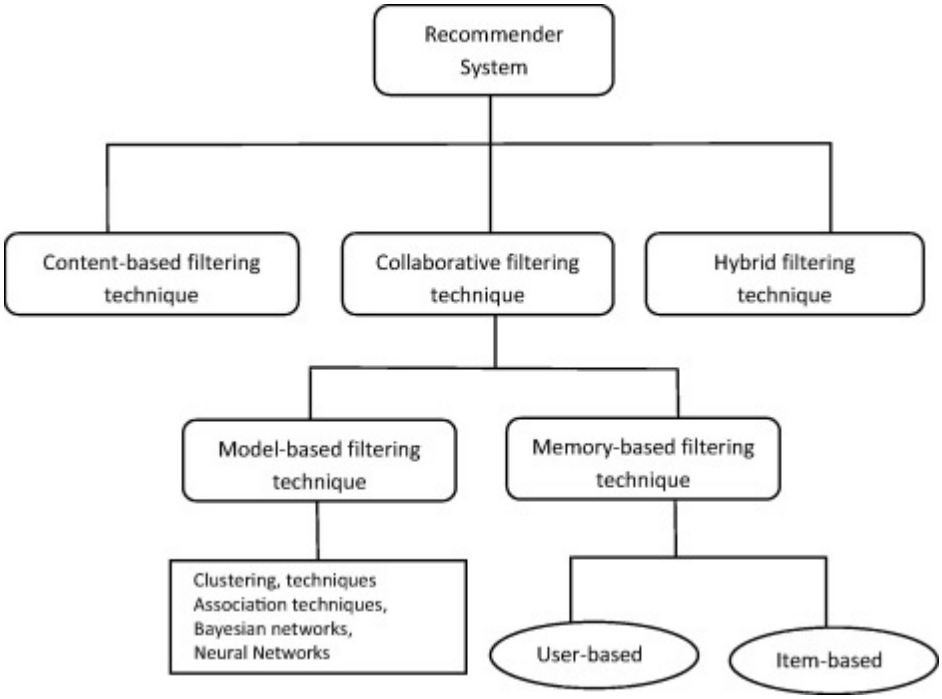


YouTube

The YouTube online video community uses Recommendation systems to create personalized recommendation so users can quickly and easily find videos that are relevant to their interests. Because of the value of keeping users engages, each user's activity on the site and to simultaneously highlight the wide range of available content. Recommendations now drive **70 percent** of overall "watch time" on YouTube, compared with **40 percent** in early 2014

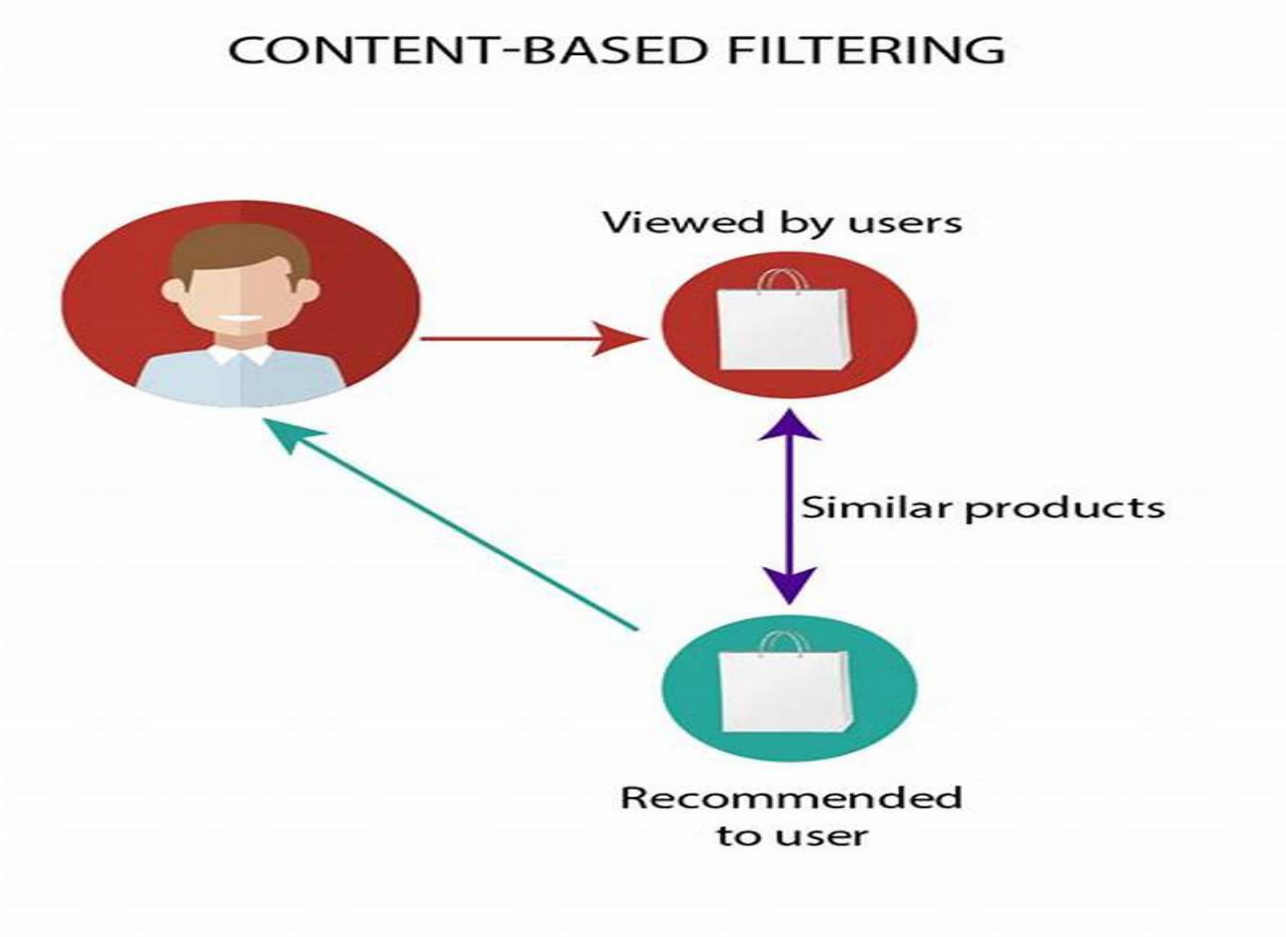


Type of recommendation



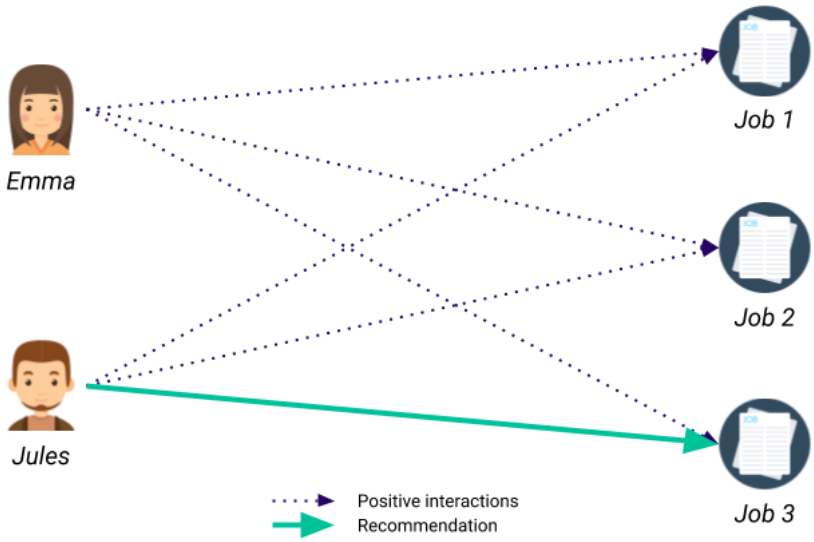
Content-based filtering

Content-based technique is a domain-dependent algorithm and it emphasizes more on the analysis of the attributes of items in order to generate predictions. When documents such as web pages, publications and news are to be recommended, content-based filtering technique is the most successful. In content-based filtering technique, recommendation is made based on the user profiles using features extracted from the content of the items the user has evaluated in the past . Items that are mostly related to the positively rated items are recommended to the user. CBF uses different types of models to find similarity between documents in order to generate meaningful recommendations. It could use Vector Space Model such as Term Frequency Inverse Document Frequency (TF/IDF) or Probabilistic models such as Naïve Bayes Classifier , Decision Trees or Neural Networks [to model the relationship between different documents within a corpus. These techniques make recommendations by learning the underlying model with either statistical analysis or machine learning techniques. Content-based filtering technique does not need the profile of other users since they do not influence recommendation. Also, if the user profile changes, CBF technique still has the potential to adjust its recommendations within a very short period of time. The major disadvantage of this technique is the need to have an in-depth knowledge and description of the features of the items in the profile.



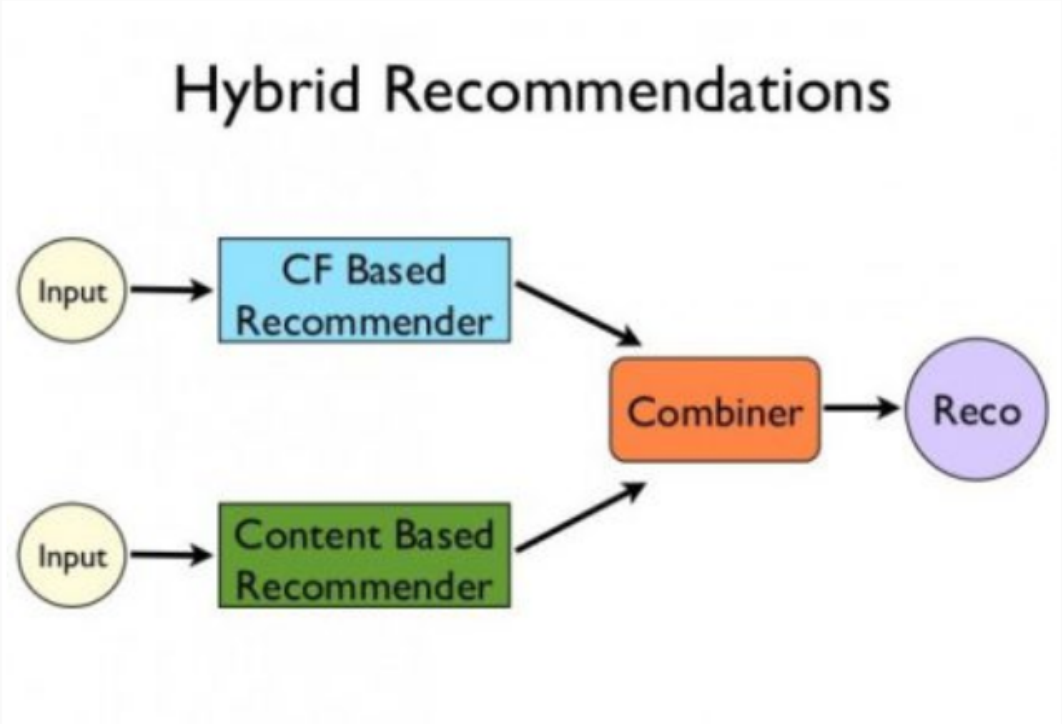
Collaborative filtering

Collaborative filtering is a domain-independent prediction technique for content that cannot easily and adequately be described by metadata such as movies and music. Collaborative filtering technique works by building a database (user-item matrix) of preferences for items by users. It then matches users with relevant interest and preferences by calculating similarities between their profiles to make recommendations Such users build a group called neighborhood. An user gets recommendations to those items that he has not rated before but that were already positively rated by users in his neighborhood. Recommendations that are produced by CF can be of either prediction or recommendation. Prediction is a numerical value, Rij, expressing the predicted score of item j for the user i, while Recommendation is a list of top N items that the user will like the most as shown in The technique of collaborative filtering can be divided into two categories: memory-based and model-based .



Hybrid filtering

Hybrid filtering technique combines different recommendation techniques in order to gain better system optimization to avoid some limitations and problems of pure recommendation systems . The idea behind hybrid techniques is that a combination of algorithms will provide more accurate and effective recommendations than a single algorithm as the disadvantages of one algorithm can be overcome by another algorithm . Using multiple recommendation techniques can suppress the weaknesses of an individual technique in a combined model. The combination of approaches can be done in any of the following ways: separate implementation of algorithms and combining the result, utilizing some content-based filtering in collaborative approach, utilizing some collaborative filtering in content-based approach, creating a unified recommendation system that brings together both approaches.


Hybrid Recommendations
Input
CF Based Recommender
Combiner
Reco
Input
Content Based Recommender

ORGANIZATION OF THE THESIS

- An implementation *recommendation system* was developed based on the weighted rates
- The focus of this *thesis* lies on user modeling for *recommender system*

THEORETICAL BACKGROUND

WEIGHTED MEAN

A weighted mean is a kind of average. Instead of each data point contributing equally to the final mean, some data points contribute more “weight” than others. If all the weights are equal, then the weighted mean equals the arithmetic mean (the regular “average” you’re used to). Weighted means are very common in statistics, especially when studying populations.

The weighted mean is relatively easy to find. But in *some* cases the weights might not add up to 1. In those cases, you’ll need to use the weighted mean formula. The only difference between the formula and the steps above is that you divide by the sum of all the weights.

$$\bar{x} = \frac{\sum_{i=1}^{n} (x_i * w_i)}{\sum_{i=1}^{n} w_i}$$

DEVELOPMENT ENVIRONMENT

DEPENDENCIES

- Python
- Numpy
- Pandas
- Ast
- Jupyter notebook

INSTALL DEPENDENCIES(Linux OS (Ubuntu 20.04 LTS)

Commands for installing pip package manager for python3

```
Step 1: $ sudo apt-get -y install python3-pip
Step 2: $ sudo pip3 install --upgrade pip
Step 3: $ pip3 –version
```

Commands for installing all dependencies are given below.

```
jupyter notebook:  $ sudo -H pip install jupyter
numpy:             $ sudo pip install numpy
Pandas:            $ sudo pip install pandas
ast:               $ sudo pip install litereval
```

DATA SET

Details about Dataset

We have used movies_metadata.csv dataset in order to build movie recommendation engine.

Data Set description

It contains 100004 ratings and 1296 tag applications across 9125 movies. These data were created by 671 users between January 09, 1995 and October 16, 2016. This dataset was generated on October 17, 2016.

Download dataset

If you want to download movies_metadata.csv dataset from its official website then use kaggle

LINK - https://www.kaggle.com/rounakbanik/the-movies-dataset?select=movies_metadata.csv

SYSTEM DESIGN AND IMPLEMENTATION

IMPORT OUR DEPENDENCIES

```
In [1]: import pandas as pd
        import numpy as np
        import ast
        from ast import literal_eval
```

LOAD DATA SET

```
In [2]: md = pd.read_csv('movies_metadata.csv')
```

UNDERSTAND DATA SET

METADATA DATAFRAME

```
In [3]: md.columns
```

```
Index(['adult', 'belongs_to_collection', 'budget', 'genres', 'homepage', 'id',
       'imdb_id', 'original_language', 'original_title', 'overview',
       'popularity', 'poster_path', 'production_companies',
       'production_countries', 'release_date', 'revenue', 'runtime',
       'spoken_languages', 'status', 'tagline', 'title', 'video',
       'vote_average', 'vote_count'],
      dtype='object')
```

FEATURES

adult: Indicates if the movie is X-Rated or Adult.

belongs_to_collection: A stringified dictionary that gives information on the movie series the particular film belongs to.

budget: The budget of the movie in dollars.

genres: A stringified list of dictionaries that list out all the genres associated with the movie.

homepage: The Official Homepage of the move.

id: The ID of the movie.

imdb_id: The IMDB ID of the movie.

original_language: The language in which the movie was originally shot in.

original_title: The original title of the movie.

overview: A brief blurb of the movie.

popularity: The Popularity Score assigned by TMDB.

poster_path: The URL of the poster image.

production_companies: A stringified list of production companies involved with the making of the movie.

production_countries: A stringified list of countries where the movie was shot/produced in.

release_date: Theatrical Release Date of the movie.

revenue: The total revenue of the movie in dollars.

runtime: The runtime of the movie in minutes.

spoken_languages: A stringified list of spoken languages in the film.

status: The status of the movie (Released, To Be Released, Announced, etc.)

tagline: The tagline of the movie.

title: The Official Title of the movie.

video: Indicates if there is a video present of the movie with TMDB.

vote_average: The average rating of the movie.

vote_count: The number of votes by users, as counted by TMDB.

In [4]:
```
md.shape
```

```
(45466, 24)
```

In [5]:
```
md.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45466 entries, 0 to 45465
Data columns (total 24 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   adult                  45466 non-null  object 
 1   belongs_to_collection  4494 non-null   object 
 2   budget                 45466 non-null  object 
 3   genres                 45466 non-null  object 
 4   homepage               7782 non-null   object 
 5   id                     45466 non-null  object 
 6   imdb_id                45449 non-null  object 
 7   original_language      45455 non-null  object 
 8   original_title         45466 non-null  object 
 9   overview               44512 non-null  object 
 10  popularity             45461 non-null  object 
 11  poster_path            45080 non-null  object 
 12  production_companies   45463 non-null  object 
 13  production_countries   45463 non-null  object 
 14  release_date           45379 non-null  object 
 15  revenue                45460 non-null  float64
 16  runtime                45203 non-null  float64
 17  spoken_languages       45460 non-null  object 
 18  status                 45379 non-null  object 
 19  tagline                20412 non-null  object 
 20  title                  45460 non-null  object 
 21  video                  45460 non-null  object 
 22  vote_average           45460 non-null  float64
 23  vote_count             45460 non-null  float64
dtypes: float64(4), object(20)
memory usage: 8.3+ MB
```

BUILD RECOMMENDATION SYSTEM

Simple recommendation system

Approach

- The Simple Recommender offers generalized recommendations to every user based on movie popularity and (sometimes) genre.

- The basic idea behind this recommender is that movies that are more popular and more critically acclaimed will have a higher probability of being liked by the average audience.

- This model does not give personalized recommendations based on the user.

What we are actually doing?

➢ The implementation of this model is extremely trivial.

➢ All we have to do is sort our movies based on ratings and popularity and display the top movies of our list.

➢ As an added step, we can pass in a genre argument to get the top movies of a particular genre.

➔ I will build our overall Top 250 Chart and will define a function to build charts for a particular genre. Let's begin!

In [6]:
```
md['genres'] = md['genres'].fillna('[]').apply(literal_eval).apply(lambda x: [i[
    'name'] for i in x] if isinstance(x, list) else [])
```

WEIGHTED RATING (WR)

v is the number of votes for the movie

m is the minimum votes required to be listed in the chart

R is the average rating of the movie

C is the mean vote across the whole report

$$Weighted\ Rating(WR) = (\frac{v}{v+m}.R) + (\frac{m}{v+m}.C)$$

```
In [7]: vote_counts = md[md['vote_count'].notnull()]['vote_count'].astype('int')
```

```
In [8]: vote_averages = md[md['vote_average'].notnull()]['vote_average'].astype('int')

        # this is C
        C = vote_averages.mean()
        C
```

```
5.244896612406511
```

- The next step, we need to determine an appropriate value for m, the minimum votes required to be listed in the chart.

- We will use 95th percentile as our cutoff. In other words, for a movie to feature in the charts, it must have more votes than at least 95% of the movies in the list.

In [9]:
```
m = vote_counts.quantile(0.95)
m
```

```
434.0
```

➢ Pre-processing step for getting year from date by splliting it using '-'

In [10]:
```
md['year'] = pd.to_datetime(md['release_date'], errors='coerce').apply(
    lambda x: str(x).split('-')[0] if x != np.nan else np.nan)
```

In [11]:
```
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
```

```
(2274, 6)
```

➢ Therefore, to qualify to be considered for the chart, a movie has to have at least 434 votes on TMDB.

➢ We also see that the average rating for a movie on TMDB is 5.244 on a scale of 10.

In [12]:
```
def weighted_rating(x):
    v = x['vote_count']
    R = x['vote_average']
    return (v/(v+m) * R) + (m/(m+v) * C)
```

In [13]:
```
qualified['wr'] = qualified.apply(weighted_rating, axis=1)
```

In [14]:
```
qualified = qualified.sort_values('wr', ascending=False).head(250)
```

```
In [15]: qualified.head(15)
```

	title	year	vote_count	vote_average	popularity	genres	wr
15480	Inception	2010	14075	8	29.108149	[Action, Thriller, Science Fiction, Mystery, A...	7.917588
12481	The Dark Knight	2008	12269	8	123.167259	[Drama, Action, Crime, Thriller]	7.905871
22879	Interstellar	2014	11187	8	32.213481	[Adventure, Drama, Science Fiction]	7.897107
2843	Fight Club	1999	9678	8	63.869599	[Drama]	7.881753
4863	The Lord of the Rings: The Fellowship of the Ring	2001	8892	8	32.070725	[Adventure, Fantasy, Action]	7.871787
292	Pulp Fiction	1994	8670	8	140.950236	[Thriller, Crime]	7.868660
314	The Shawshank Redemption	1994	8358	8	51.645403	[Drama, Crime]	7.864000
7000	The Lord of the Rings: The Return of the King	2003	8226	8	29.324358	[Adventure, Fantasy, Action]	7.861927
351	Forrest Gump	1994	8147	8	48.307194	[Comedy, Drama, Romance]	7.860656
5814	The Lord of the Rings: The Two Towers	2002	7641	8	29.423537	[Adventure, Fantasy, Action]	7.851924
256	Star Wars	1977	6778	8	42.149697	[Adventure, Action, Science Fiction]	7.834205
1225	Back to the Future	1985	6239	8	25.778509	[Adventure, Comedy, Science Fiction, Family]	7.820813
834	The Godfather	1972	6024	8	41.109264	[Drama, Crime]	7.814847
1154	The Empire Strikes Back	1980	5998	8	19.470959	[Adventure, Action, Science Fiction]	7.814099
46	Se7en	1995	5915	8	18.45743	[Crime, Mystery, Thriller]	7.811669

➢ We see that three Christopher Nolan Films, Inception, The Dark Knight and Interstellar occur at the very top of our chart.

➢ The chart also indicates a strong bias of TMDB Users towards particular genres and directors.

➢ Let us now construct our function that builds charts for particular genres.

➢ For this, we relax our default conditions to the 85th percentile instead of 95.

In [16]:
```
'''
>>> s
     a   b
one  1.  2.
two  3.  4.

>>> s.stack()
one a    1
    b    2
two a    3
    b    4
'''
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)
gen_md.head(3).transpose()
```

	0	0	
adult	False	False	False
belongs_to_collection	{'id': 10194, 'name': 'Toy Story Collection', ...	{'id': 10194, 'name': 'Toy Story Collection', ...	{'id': 10194, 'name': 'Toy Story Collection', ...
budget	30000000	30000000	30000000
homepage	http://toystory.disney.com/toy-story	http://toystory.disney.com/toy-story	http://toystory.disney.com/toy-stor
id	862	862	862
imdb_id	tt0114709	tt0114709	tt0114709
original_language	en	en	en
original_title	Toy Story	Toy Story	Toy Story
overview	Led by Woody, Andy's toys live happily in his ...	Led by Woody, Andy's toys live happily in his ...	Led by Woody, Andy's toys live happily in his ...
popularity	21.946943	21.946943	21.946943
poster_path	/rhIRbceoE9lR4veEXuwCC2wARtG.jpg	/rhIRbceoE9lR4veEXuwCC2wARtG.jpg	/rhIRbceoE9lR4veEXuwCC2wARtG.
production_companies	[{'name': 'Pixar Animation Studios', 'id': 3}]	[{'name': 'Pixar Animation Studios', 'id': 3}]	[{'name': 'Pixar Animation Studios' 'id': 3}]
production_countries	[{'iso_3166_1': 'US', 'name': 'United States o...	[{'iso_3166_1': 'US', 'name': 'United States o...	[{'iso_3166_1': 'US', 'name': 'Unite States o...
release_date	1995-10-30	1995-10-30	1995-10-30
revenue	373554033.0	373554033.0	373554033.0
runtime	81.0	81.0	81.0
spoken_languages	[{'iso_639_1': 'en', 'name': 'English'}]	[{'iso_639_1': 'en', 'name': 'English'}]	[{'iso_639_1': 'en', 'name': 'English
status	Released	Released	Released
tagline	NaN	NaN	NaN
title	Toy Story	Toy Story	Toy Story
video	False	False	False
vote_average	7.7	7.7	7.7
vote_count	5415.0	5415.0	5415.0
year	1995	1995	1995
genre	Animation	Comedy	Family

In [17]:

```
def build_chart(genre, percentile=0.85):
    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',
    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_
                        axis=1)
    qualified = qualified.sort_values('wr', ascending=False).head(250)

    return qualified
```

EXPERIMENTAL RESULTS AND ANALYSIS

➔ Let us see our method in action by displaying the Top 15 Romance Movies (Romance almost didn't feature at all in our Generic Top Chart despite being one of the most popular movie genres).

Top 15 Romantic Movies

```
In [18]: build_chart('Romance').head(15)
```

	title	year	vote_count	vote_average	popularity	wr
10309	Dilwale Dulhania Le Jayenge	1995	661	9	34.457024	8.565285
351	Forrest Gump	1994	8147	8	48.307194	7.971357
876	Vertigo	1958	1162	8	18.20822	7.811667
40251	Your Name.	2016	1030	8	34.461252	7.789489
883	Some Like It Hot	1959	835	8	11.845107	7.745154
1132	Cinema Paradiso	1988	834	8	14.177005	7.744878
19901	Paperman	2012	734	8	7.198633	7.713951
37863	Sing Street	2016	669	8	10.672862	7.689483
882	The Apartment	1960	498	8	11.994281	7.599317
38718	The Handmaiden	2016	453	8	16.727405	7.566166
3189	City Lights	1931	444	8	10.891524	7.558867
24886	The Way He Looks	2014	262	8	5.711274	7.331363
45437	In a Heartbeat	2017	146	8	20.82178	7.003959
1639	Titanic	1997	7770	7	26.88907	6.981546
19731	Silver Linings Playbook	2012	4840	7	14.488111	6.970581

Top 6 Action Movies

```
In [19]: build_chart('Action').head(6)
```

	title	year	vote_count	vote_average	popularity	wr
15480	Inception	2010	14075	8	29.108149	7.955099
12481	The Dark Knight	2008	12269	8	123.167259	7.948610
4863	The Lord of the Rings: The Fellowship of the Ring	2001	8892	8	32.070725	7.929579
7000	The Lord of the Rings: The Return of the King	2003	8226	8	29.324358	7.924031
5814	The Lord of the Rings: The Two Towers	2002	7641	8	29.423537	7.918382
256	Star Wars	1977	6778	8	42.149697	7.908327

Top 3 Family Movies

In [20]:
```
build_chart('Family').head(3)
```

	title	year	vote_count	vote_average	popularity	wr
1225	Back to the Future	1985	6239	8	25.778509	7.893053
359	The Lion King	1994	5520	8	21.605761	7.879754
5481	Spirited Away	2001	3968	8	41.048867	7.835635

Top 8 Comedy Movies

In [21]:
```
build_chart('Comedy').head(8)
```

	title	year	vote_count	vote_average	popularity	wr
10309	Dilwale Dulhania Le Jayenge	1995	661	9	34.457024	8.463024
351	Forrest Gump	1994	8147	8	48.307194	7.963363
1225	Back to the Future	1985	6239	8	25.778509	7.952358
18465	The Intouchables	2011	5410	8	16.086919	7.945207
22841	The Grand Budapest Hotel	2014	4644	8	14.442048	7.936384
2211	Life Is Beautiful	1997	3643	8	39.39497	7.919430
732	Dr. Strangelove or: How I Learned to Stop Worr...	1964	1472	8	9.80398	7.809073
3342	Modern Times	1936	881	8	8.159556	7.695554

Top 12 Animation Movies

In [22]:
```
build_chart('Animation').head(12)
```

	title	year	vote_count	vote_average	popularity	wr
359	The Lion King	1994	5520	8	21.605761	7.909339
5481	Spirited Away	2001	3968	8	41.048867	7.875933
9698	Howl's Moving Castle	2004	2049	8	16.136048	7.772103
2884	Princess Mononoke	1997	2041	8	17.166725	7.771305
5833	My Neighbor Totoro	1988	1730	8	13.507299	7.735274
40251	Your Name.	2016	1030	8	34.461252	7.589820
5553	Grave of the Fireflies	1988	974	8	0.010902	7.570962
19901	Paperman	2012	734	8	7.198633	7.465676
39386	Piper	2016	487	8	11.243161	7.285132
20779	Wolf Children	2012	483	8	10.249498	7.281198
25044	Song of the Sea	2014	420	8	6.967358	7.212999
31658	Feast	2014	420	8	7.365663	7.212999

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. This project discussed the recommendation techniques and highlighted their strengths and challenges with diverse kind of ***weighted rating*** *strategies used to improve their performances. Various learning algorithms used in generating recommendation models.*

REFERENCES

https://github.com/RAGUPATHI-M/RECOMMENDATION-SYSTEM