

A  
Mini Project Report

On

**MOVIE RECOMMENDER SYSTEM**

Submitted to

**RAJIV GANDHI UNIVERSITY OF KNOWLEDGE AND TECHNOLOGIES  
RK VALLEY**

*in partial fulfilment of the requirement for the award of the Degree  
of*

**BACHELOR OF TECHNOLOGY**

In

**COMPUTER SCIENCE & ENGINEERING**

Submitted by

**D.Karishma (R180458)**

**D.Lakshmi Priyanka (R180851)**

Under the Guidance of

**Mr. N.Satyanandaram, Lecturer in Dept.of CSE**



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**RAJIV GANDHI UNIVERSITY OF KNOWLEDGE  
TECHNOLOGIES**

**(catering the Educational Needs of Gifted Rural Youth of AP)**

**R.K Valley,Vempalli(M), Kadapa(Dist) – 516330**

# **RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES**

**(A.P.Government Act 18 of 2008)**

**RGUKT-RK Valley**

**Vempalli, Kadapa, Andhrapradesh-516330.**

## **CERTIFICATE OF PROJECT COMPLETION**

This is to certify that I have examined the thesis entitled “**MOVIE RECOMMENDER SYSTEM**” submitted by **D.Karishma (R180458),D.Lakshmi priyanka (R180851)** under our guidance and supervision for the partial fulfilment for the degree of Bachelor of Technology in computer Science and Engineering during the academic session JULY 2023 – MAY 2024 at RGUKT-RKVALLEY.

### **Project Guide**

Mr. N.Satyanandaram,  
Lecturer. in Dept of CSE,  
RGUKT-RK Valley.

### **Head of the Department**

Mr. N.Satyanandaram,  
Lecturer in Dept of CSE,  
RGUKT-RK Valley.

# **RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES**

**(A.P.Government Act 18 of 2008)**

**RGUKT-RK Valley**

**Vempalli, Kadapa, Andhrapradesh-516330.**

## **DECLARATION**

We,**D.Karishma(R180458),D.LakshmiPriyanka(R180851)** here by declare that the project report entitled “**MOVIE RECOMMENDER SYSTEM**” done under the guidance of **Mr. N.Satyanandaram** fulfillment for the degree of Bachelor of Technology in Computer Science and Engineering during the academic session **July 2023-May 2024** at **RGUKT-RK Valley**.We also declare that this project is a result of our own effort and has not been copied or imitated from any websites are mentioned in the references.To the best of our knowledge,the results embodied in this disertation work have not been submittes to any universityhave not been submitted to any university or institute for the award of any degree or diploma.

**D.KARISHMA(R180458)**

**D.LAKSHMI PRIYANKA(R180851)**

## ACKNOWLEDGEMENT

we would like to express my deep sense of gratitude & respect to all those people behind the screen who guided, inspired and helped us crown all our efforts with success. we wish to express our gratitude to **Mr. N.Satyanandaram** for his valuable guidance at all stages of study, advice, constructive suggestions, supportive attitude and continuous encouragement, without which it would not be possible to complete this project.

I would also like to extend our deepest gratitude & reverence to the Director of RGUKT, RK Valley **Prof. A V S S KUMARA SWAMI GUPTHA** and HOD of Computer Science and Engineering **Mr. N. Satyanandaram** for their constant support and encouragement.

Last but not least I express my gratitude to my parents for their constant source of encouragement and inspiration for me to keep my morals high.

**With Sincere Regards,**

**D.Karishma,**

**R180458.**

**D.L.Priyanka**

**R180851.**

# ABSTRACT

---

Nowadays, there has been immense growth in the amount of digital information and the number of users as well. Also, the quantity of data transactions on the internet has drastically increased. This triggered a potential challenge of information overload which obstructs timely access to the available data on the internet. With the number of users increasing, the amount of data is increasing too dramatically. The problem arises when a user has to search for long periods to obtain their desired movie or information. The recommendation system helps in solving this problem. The recommendation system helps the user to find their desirable item by predicting their previous performance and suggesting relevant information to the user. They are essentially a central part of websites like movies, music, e-commerce applications, and many more.

In today's world of internet, Recommendation systems play a major role in our day to day life by providing suggestions to users for certain resources like movies, books, education, online shopping, etc. It has the ability to determine whether a user would prefer an item or not, based on the user's profile. A recommendation system is one of the widespread applications of machine learning that deals with the tendency of a certain user towards an item based on his/her likeliness towards it previously. Many factors can be considered while developing a movie recommender system like genre, ratings, tags, feedback, director of the movie, and many more.

In this project, recommendation systems for the movies are developed. Movies recommendation systems usually predict what movies a user will like based on the attributes related to the previous search history or liked movies. The recommender movie system can recommend a movie by a combination of one, two, or more attributes. The proposed system helps the user in picking movies, where the users search input genre, ratings, and tags would help to predict the movies for that particular user. The approach adopted to do so is content-based filtering using genre correlation.

# TABLE OF CONTENTS

Project title.....	
Project completion certificate.....	
Declaration.....	
Acknowledgment.....	
Abstract.....	
Table of contents.....	
List of abbreviations.....	
1.INTRODUCTION.....	
1.1 Overview	
1.2 Background and motivation	
1.3 Problem statement	
1.4 Objective	
1.5 Scope and limitation	
1.6 Development methodology	
2.BACKGROUND AND RESEARCH.....	
2.1 Background study	
2.2 Supervised learning	
2.3 Literaturereview	
2.4 Content-Based filtering	
2.5 Existing system	
3.SPECIFICATION AND DESIGN.....	
3.1 Requirement elicitation and Analysis	
3.1.1 Functional requirement	
3.1.2 Non-Functional requirement	

3.1.3 Hardware requirements	
3.1.4 Software requirements	
3.2 Feasibility analysis	
3.2.1 Technical feasibility	
3.2.2 Operational feasibility	
3.2.3 Economical feasibility	
3.3 Analysis	
3.3.1 Class diagram	
3.3.2 Activity diagram	
3.3.3 Entities and attributes	
3.3.4 System flow chart	
3.3.5 Sequence diagram	
4.SYSTEM DESIGN.....	
4.1 DFD diagram	
4.2 Vectorization	
4.3 NLTK	
5.IMPLEMENTATION.....	
5.1 Code implementation	
5.2 Dataset	
5.3 Tools used	
5.4 Output	
6.CONCLUSION AND FUTURE SCOPE.....	
6.1 Conclusion	
6.2 Future scope	
6.3 References	

## **LIST OF ABBREVIATIONS**

App	Application
ID	Identification
ML	Machine Learning
Numpy	Numerical Python
PIP	Preferred Installer Program
TP	True Positive
WBS	Work breakdown structure



# **1.INTRODUCTION**

## **1.1. Overview**

In this project, we recommend movies to the users based on the history of their personalized searches and reviews. Both people and online streaming services need a user based personalized movie recommendation system which helps in analyzing the interests of each individual and recommends the best movies suited to them. The approach used for building the movie recommendation system is content-based filtering. As we know that content-based filtering analyses the user's past behavior and recommends items similar to it based on the parameters considered. In this Content-based movie recommender system gives their recommendations based on attributes like title, Genre, and Casts.

## **1.2. Background and Motivation**

Recommender systems are based on a variety of approaches such as content based, collaborative approach hybrid. Furthermost movie recommendation systems are centered on collaborative filtering and clustering. In movie recommender systems the user is asked to rate the movies which user has already seen then these ratings are applied to recommend other movies to the user that user has not perceived by utilizing collaborative filtering that is based on similar ratings. Collaborative filtering is tremendously spreading in such a way that this approach influences most of the recommender systems. Collaborative filtering majorly classified into two principal classes such as memory-based collaborative filtering and model based collaborative filtering. Memory-based collaborative filtering explores for nearest neighbors in the user space for an active user.

## **1.3. Problem Statement**

Movie recommendation system provides related content out of relevant and irrelevant collection of items to users of online service providers. Online Movie Recommendation System aims to recommend movies to users based on user-movie item, genre, cast and previous searches or preferences. It is a web application that allows users to search movies as well as it recommends them appropriate movies based on their search patterns

## **1.4. Objective**

The Movie Recommendation System provides a mechanism to help users categorize users with similar interests. Basically the purpose of a recommendation system is to search for material that will be interesting to a person. Moreover, it involves a number of factors to create personalized

lists of useful and interesting content specific to each user/individual.

Recommendation algorithm collects from the user interest. A set of keywords (or features) of an item is the Item profile. For example, consider a scenario in which a person goes to buy his favorite cake 'X' to a pastry. Unfortunately, cake 'X' has been sold out and as a result of this the shopkeeper recommends the person to buy cake 'Y' which is made up of ingredients similar to cake 'X'.

## **1.5. Scope and Limitation**

### **Work on several numbers of data:**

The number of choices for anything on internet is very high and it's tedious to refine most wanted data by self while searching. The scope of this proposal system includes working within numerous data, with ease.

### **Saving of time:**

Many people have problem selecting the alternative item of movie due to lack of time and due to search issues. Also movie recommendations from friends can be time consuming. The system helps in saving lots of time.

### **Relief from processor problem:**

Many mobile phone and limited processing power computers can't handle recommender system due to its extremely large dataset. The solution opted for this can be use of web services. The proposed system uses web services, thus makes process simpler.

## **1.6. Development Methodology**

Incremental methodology was chosen for this project as the requirements for the project are clearly known, defined as well as understood. As the methodology supports the process of design, implementation and testing with each increment added over a course of time, incremental methodology would be a great fit for the project as animal detection task needs a lot of hit and trial in order to fulfill the proposed requirements and functionalities of the aimed project as well as to achieve a good user experience. However, in the making process of the system some minor features can be added but as the main requirements are understood this methodology would be great fit.

The main requirement of the attendance system is to be able to input

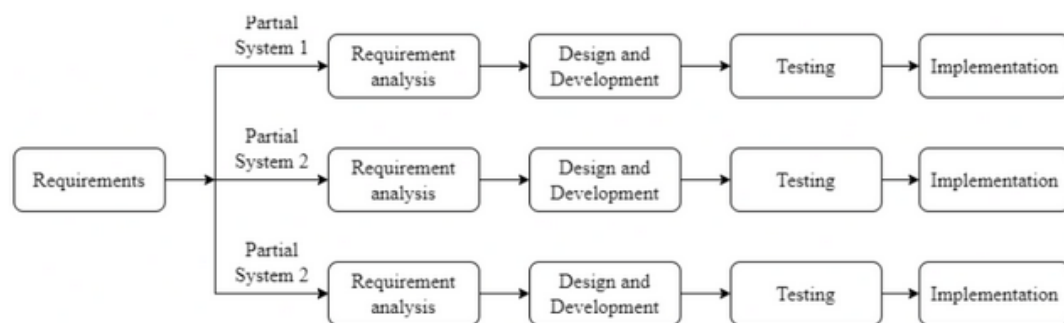
photo in the system and get classification / attendance as output.

Based on this, the system development was broken down into many partial development projects. These partial system development projects were:

- \*The development of the system that takes the text as input and passes it to the model.

- \*The system that takes a text as the input and passes it to the model.

- \*The system that takes the output generated by the model and gives a output result.



**Figure 1.6: Block Diagram of Incremental Model [1]**

## **2.BACKGROUND STUDY AND MOTIVATION**

### **2.1. Background Study**

With the various types of datasets that have been utilized for detection jobs in today's world, the detection sector has been able to accomplish significant growth. However, because real- life objects are often variable in nature due to environmental differences such as size, background, size, and many other aspects, annotation jobs can be a significant difficulty in the field of categorization and detection.

### **2.2 Supervised Learning**

There is little doubt machine learning has become one of the most powerful technologies in the last decade. The emphasis on “learning” in machine learning allows computers to make better and better decisions, based on previous experiences. Classical machine learning is often categorized into supervised, unsupervised, semi-supervised or reinforced learning depending on how the algorithm learns to become more accurate in its predictions. Among them, supervised learning is one of the heavily explored and important form of ML. In Supervised Learning, the learning process is done under the seen label of observation variables. Datasets are trained with the training sets to build a model which is used later on to label new observations or data points from the testing set. As for the training set, the input variables are the features which will influence the accuracy of predicted variable. It contains both quantitative and qualitative variables; the output variable is the label class that Supervised Learning will label the new observations.

### **2.3. Literature Review**

There are three techniques of recommendation system: Collaborative Filtering, Content-Based Filtering and Hybrid Filtering. In Content Based recommender system, use provides data either explicitly (rating) or implicitly (by clicking on a link). The system captures this data and generates user profile for every user. By making use of user profile, recommendation is generated. In content based filtering, recommendation is given by only watching single user’s profile. System tries to recommend item similar to that item based on users past activity. Unlike content based, collaborative filtering finds those users whose likings are similar to a given user. It then recommends item or any product, by considering that the given user will also like the item which other users like because their taste are

similar. Both these technique have their own strength and weakness so to overcome this, hybrid technique came into picture, which is a combination of both these techniques.

Hybrid filtering can be used in various types. We can use content based filtering first and then pass those results to collaborative recommender (and vice-versa) or by integrating both the filter into one model to generate the result. These kinds of modifications are also uses to cope up with cold start, data sparsity and scalability problem.

## **2.4 Content-Based Filtering**

Content-Based Filtering are also known as cognitive filtering. This filtering recommends item to the user based on his past experience. For example, if a user likes only action movies then the system predicts him only action movies similar to it which he has highly rated. The broader explanation could be suppose the user likes only politics related content so the system suggests the websites, blogs or the news similar to that content. Unlike collaborative filtering, content based filtering do not face new user problem. It does not have other user interaction in it. It only deals with particular user's interest. Content based filtering first checks the user preference and then suggest him with the movies or any other product to him. It only focus on single user's ideas, thoughts and give prediction based on his interest. So if we talk about movies, then the content based filtering technique checks the rating given by the user.

## **2.5. Existing System**

The reason behind this improvement is the popularity gained by organizations like Netflix whose primary objective is customer satisfaction. Before existence the recommendation system, individuals would physically choose movies to watch from movie libraries. They either had to read the user's reviews or based on the review they would select a movie or had to randomly select a movie. This procedure isn't feasible, as there is an enormous number of spectators with a unique preference for movies. Hence many recommendation systems have been developed over the past decade. These systems use different approaches like a collaborative approach, a content-based approach, a hybrid approach, etc. Taking a look at the behavior and history of different clients, based on their ratings, the system suggests to us what to watch without having to put effort into deciding what to watch. These recommendation systems are categorized into two types, i.e. collaborative filtering approach and content-based approach. The

collaborative approach combines the ratings of different users that have similar tastes and then recommends the movies whereas the content-based approach is limited to a single user, where the user's past history and ratings are used for providing recommendations. There are a number of methodologies introduced to implement this recommendation system which includes various fields of Data Mining, Clustering and Bayesian Network methodology.

## **3.SPECIFICATIONS AND DESIGN**

### **3.1. Requirement Elicitation and Analysis**

Movie recommendation system is a web-based app, which provides all the details of the requested movie. Details include recommended movies, as well as top cast, ratings, reviews and so on. The requirement of this project is given below:

#### **3.1.1. Functional Requirement**

- Validate each user input to database.
- Autosuggest user for smooth experience.
- Simple loading screen to inform user that work is in progress.
- Notify user if result is not found.
- Simply UI to show more details about the casts.

#### **3.1.2. Non-Functional Requirement**

- The processing of each request should be done around 10 seconds.
- Display default data if some data are missing.
- Search result, although autosuggest unable to suggest.

#### **3.1.3. Hardware Requirements**

- Processor: Pentium IV or higher
- RAM: 256 MB
- Network: Bandwidth greater than 50 KBps (400 kbps)

#### **3.1.4 Software Requirements**

- OS: Windows, iOS, Linux
- Browser: Chrome, Brave, Mozilla Firefox, Microsoft Edge, Apple Safari, Opera

### **3.2 Feasibility Analysis**

#### **3.2.1. Technical Feasibility**

In this project, the system was implemented as a web-based application. The primary programming language selected to build the system was Python (version 3.6.5), which is an open-source programming language. The programming language was selected due its ease of use and

my experience in working with the programming language. Internal technical capabilities are required to support the project requirements that show the project technically feasible to work upon. Except for python ,Jupyter notebook,Streamlit technologies will be used in this project, which are one of the most feasible technologies in web development.

The tools, systems, modules and libraries needed to build the system are all open source, freely available and are easy to use. Hence, the project was determined technically feasible.

### **3.2.2. Operational Feasibility**

The system built in the project follows a client-server architecture. Training of the model,which is resource-intensive and time consuming, is a one-time process and need not be done repeatedly as long as the model holds true for future data acquisition. The well-planned design of the system will ensure the optimal utilization of the computer resources (such as storage, memory processing, etc). The system will have simple user interface, so that any non-technical user can operate with the system. As the hardware required for operation of the system is readily available and fairly inexpensive, the product is deemed operationally feasible.

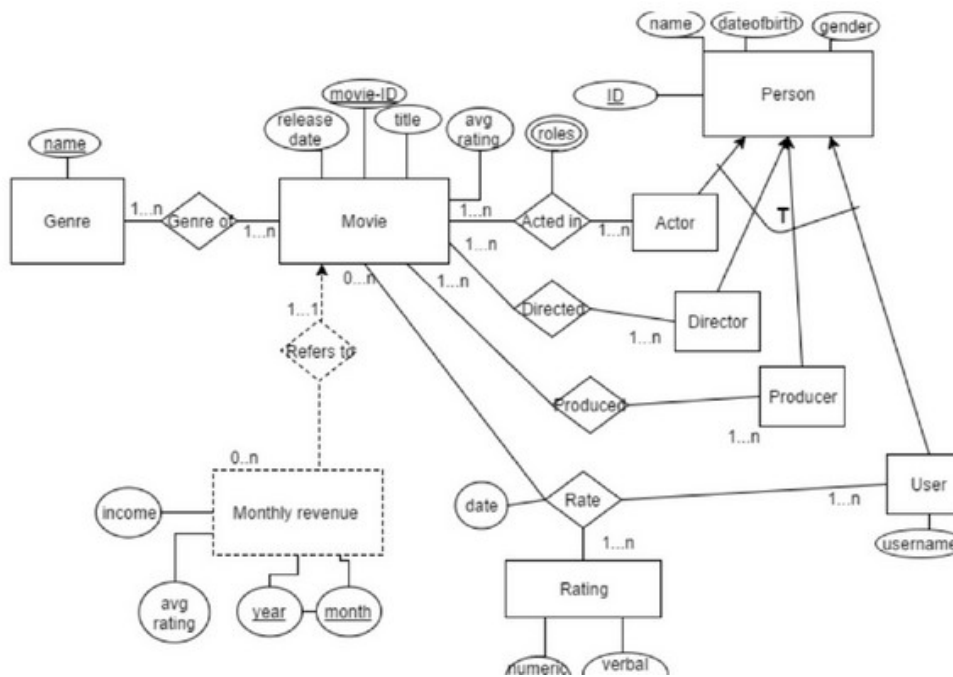
### **3.2.3. Economic Feasibility**

Personal computer, personal mobile and personal internet subscription was used for the majority of the project. This system is very economically feasible it will not require the budget of huge, amount to create the entire system. As it will not contain the large database and similarly the data will not necessarily require being stored permanently forever so it will become somehow easy to maintain the entire system and the maintenance cost are not over headed. Once the system is ready and user gets guidelines to use it, it is the easiest and very low cost system to implement.

## **3.3 Analysis**

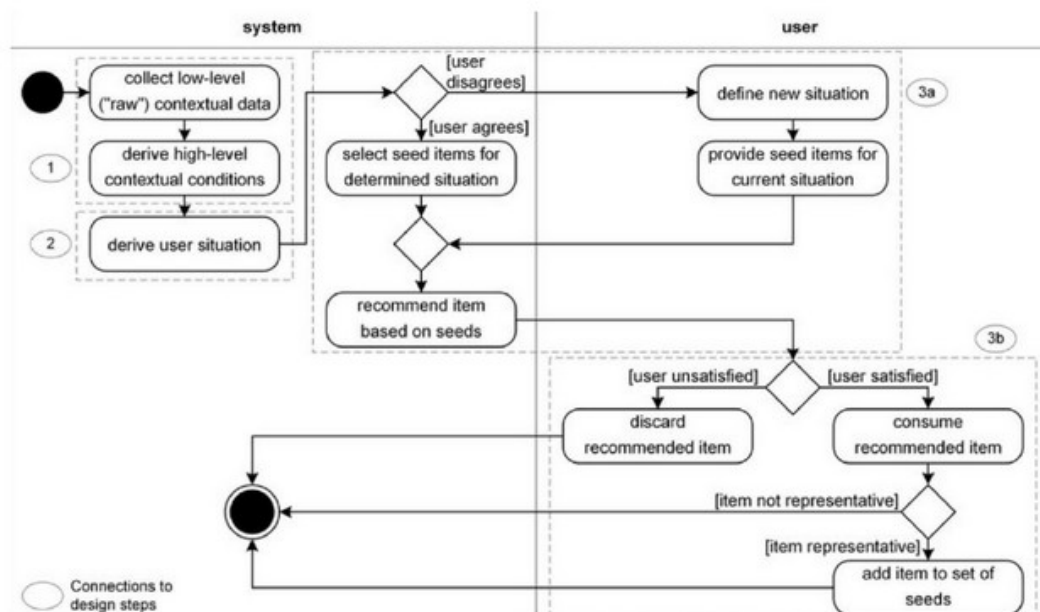
### **3.3.1 Class diagram**





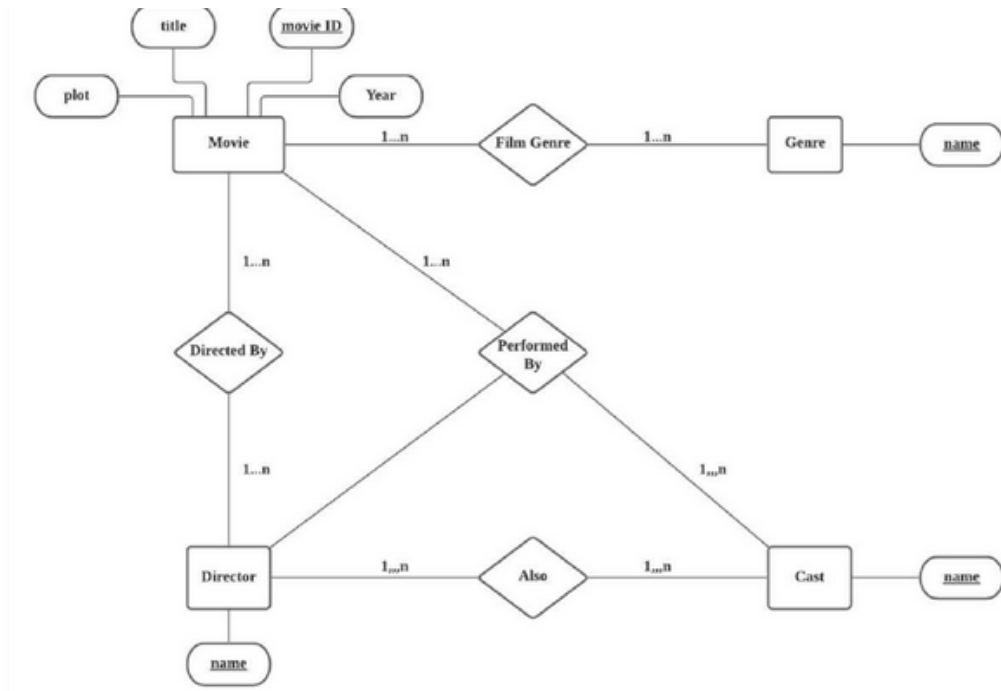
CLASS DIAGRAM FOR MOVIE RECOMMENDED SYSTEM

### 3.3.2 Activity Diagram



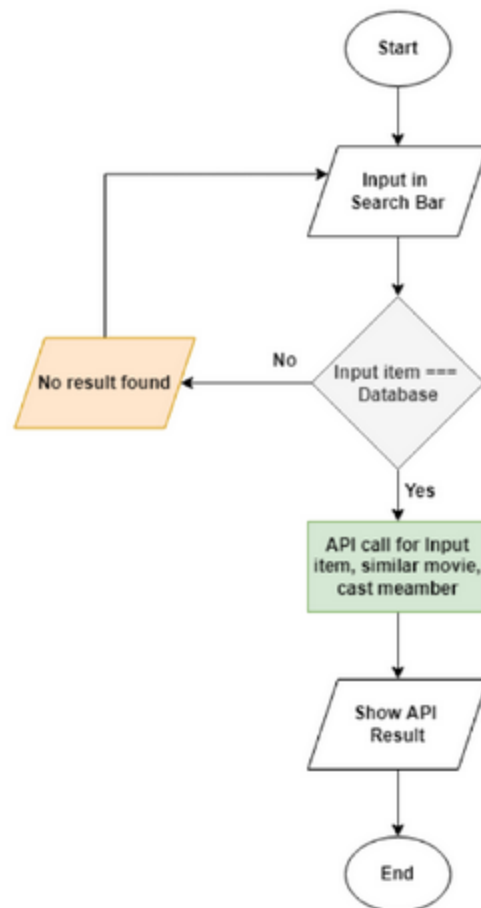
ACTIVITY DIAGRAM FOR MOVIE RECOMMENDED SYSTEM

### 3.3.3 Entities and Attributes

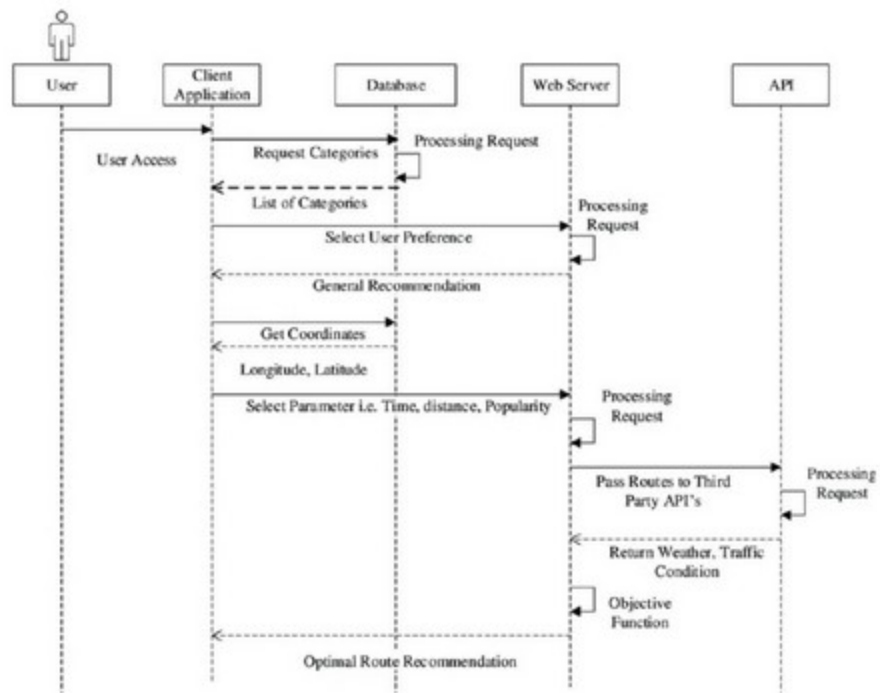


## ER DIAGRAM

### 3.3.4 System Flow Chart



### 3.3.5 Sequence diagram

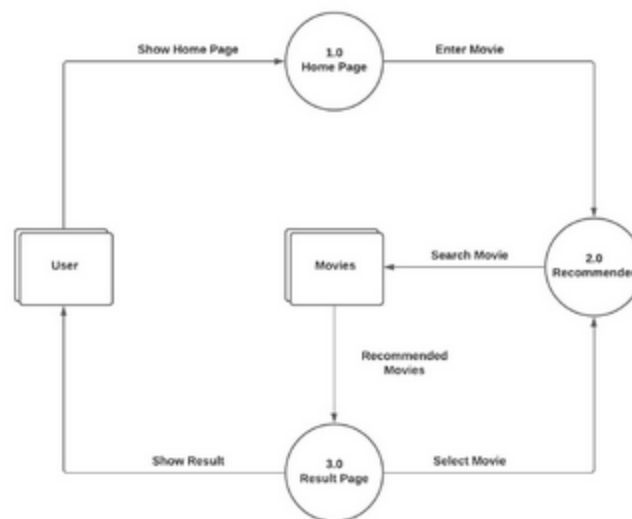


## 4.SYSTEM DESIGN

### 4.1 DFD Diagram



Fig: DFD Level-0



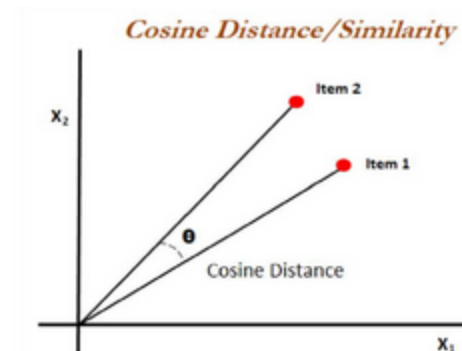
## 4.2 Vectorization

### Similarity Score

It is a numerical value ranges between zero to one which helps to determine how much two items are similar to each other on a scale of zero to one. This similarity score is obtained measuring the similarity between the text details of both of the items. So, similarity score is the measure of similarity between given text details of two items. This can be done by cosine similarity. Cosine similarity is a metric used to measure how similar the documents are irrespective of their size. Mathematically, it measures the cosine of the angle between two vectors projected in a multi-dimensional space.

The cosine similarity is advantageous because even if the two similar documents are far apart by the Euclidean distance (due to the size of the document), chances are they may still be oriented closer together. The smaller the angle, higher the cosine similarity.

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}}$$



## 4.3 NLTK

### Natural language toolkit

NLTK is a standard python library with functions and utilities for the ease of use and implementation. It is one of the most used libraries for natural language processing and computational linguistics.

### NLTK Installation Process

With a system running windows OS and having python preinstalled

Open a command prompt and type:

```
pip install nltk
```

Note:

### **!pip install nltk**

will download nltk in a specific file/editor for the current session

### **nltk dataset download**

There are several datasets which can be used with nltk. To use them, we need to download them.

We can download them by executing this:

```
#code
```

```
import nltk
```

```
nltk.download()
```

Click download in the pop up

Once it downloads, we are set to go.

### **Accessing a dataset in NLTK**

A dataset is referred to as corpus in nltk.

A corpus is essentially a collection of sentences which serves as an input. For further processing a corpus is broken down into smaller pieces and processed which we would see in later sections.

There are several of them which we downloaded in the earlier step, but we have used the movie\_reviews corpus for the demonstration.

```
import nltk
```

```
from nltk.corpus import movie_reviews,movie_reviews.words()
```

## 5.IMPLEMENTATION

### 5.1Code Implementation

#### Importing libraries

##### Numpy

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

##### Pandas

Pandas is a software library written for the Python programming language for data manipulation and analysis.

##### SK learning

scikit-learn is a Python module for machine learning built on top of SciPy and is distributed under the 3-Clause BSD license.

##### Pickle

Pickling is a way to convert a Python object (list, dictionary, etc.) into a character stream. The idea is that this character stream contains all the information necessary to reconstruct the object in another pythonscript.

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

```
In [49]: from sklearn.feature_extraction.text import CountVectorizer  
cv = CountVectorizer(max_features=5000, stop_words='english')
```

```
In [62]: import pickle
```

#### Reading data sets,merging datasets

```
In [2]: movies = pd.read_csv('tmdb_5000_movies.csv')
credits = pd.read_csv('tmdb_5000_credits.csv')

In [3]: movies.head(1)

Out[3]:
```

budget	genres	homepage	id	keywords	original_language	original_title	overview	popularity	production_companies	production_country
000000	[{"id": 28, "name": "Action"}, {"id": 12, "name": "Adventure"}, {"id": 14, "name": "Fantasy"}, {"id": 878, "name": "Science Fiction"}]	http://www.avatarmovie.com/	19995	[{"id": 1463, "name": "culture clash"}, {"id": 12, "name": "culture clash"}, {"id": 12, "name": "culture clash"}]	en	Avatar	In the 22nd century, a paraplegic Marine is di...	150.437577	[{"name": "Ingenious Film Partners", "id": 289...}, {"name": "United States", "id": 10...}, {"name": "Twentieth Century Fox", "id": 10...}]	US

```
In [4]: credits.head(1)

Out[4]:
```

movie_id	title	cast	crew
0	19995 Avatar	[{"cast_id": 242, "character": "Jake Sully", "..."}, {"cast_id": 52, "character": "Travis Mayweather", "..."}]	[{"credit_id": "52fe48009251416c750aca23", "department": "Casting", "job": "Casting Director", "name": "Deborah A. Simon", "gender": "female", "slug": "deborah-a-simon"}]

//merging the both data

```
In [5]: movies = movies.merge(credits,on='title')

In [6]: movies.head(1)
```

## Removing duplicates ,null values and reducing columns to limited database

```
movies = movies[['movie_id','title','overview','genres','keywords','cast','crew']]

In [8]: movies.head()

Out[8]:
```

	movie_id	title	overview	genres	keywords	cast	crew
0	19995	Avatar	In the 22nd century, a paraplegic Marine is di...	[{"id": 28, "name": "Action"}, {"id": 12, "name": "Adventure"}, {"id": 14, "name": "Fantasy"}, {"id": 878, "name": "Science Fiction"}]	[{"id": 1463, "name": "culture clash"}, {"id": 12, "name": "culture clash"}, {"id": 12, "name": "culture clash"}]	[{"cast_id": 242, "character": "Jake Sully", "..."}, {"cast_id": 52, "character": "Travis Mayweather", "..."}]	[{"credit_id": "52fe48009251416c750aca23", "department": "Casting", "job": "Casting Director", "name": "Deborah A. Simon", "gender": "female", "slug": "deborah-a-simon"}]
1	285	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	[{"id": 12, "name": "Adventure"}, {"id": 14, "name": "Fantasy"}, {"id": 270, "name": "Ocean"}, {"id": 726, "name": "Romance"}]	[{"id": 270, "name": "Ocean"}, {"id": 726, "name": "Romance"}]	[{"cast_id": 4, "character": "Captain Jack Sparrow", "..."}, {"cast_id": 5, "character": "Will Turner", "..."}]	[{"credit_id": "52fe4232c3a36847f800b579", "department": "Casting", "job": "Casting Director", "name": "Deborah A. Simon", "gender": "female", "slug": "deborah-a-simon"}]
2	206647	Spectre	A cryptic message from Bond's past sends him o...	[{"id": 28, "name": "Action"}, {"id": 12, "name": "Adventure"}, {"id": 14, "name": "Fantasy"}, {"id": 878, "name": "Science Fiction"}]	[{"id": 470, "name": "spy"}, {"id": 818, "name": "Thriller"}, {"id": 853, "name": "Mystery"}]	[{"cast_id": 1, "character": "James Bond", "credit_id": "54805967c3a36829b5002c41", "department": "Casting", "job": "Casting Director", "name": "Deborah A. Simon", "gender": "female", "slug": "deborah-a-simon"}]	[{"credit_id": "54805967c3a36829b5002c41", "department": "Casting", "job": "Casting Director", "name": "Deborah A. Simon", "gender": "female", "slug": "deborah-a-simon"}]
3	49026	The Dark Knight Rises	Following the death of District Attorney Harve...	[{"id": 28, "name": "Action"}, {"id": 80, "name": "Drama"}, {"id": 853, "name": "Mystery"}, {"id": 878, "name": "Science Fiction"}]	[{"id": 849, "name": "dc comics"}, {"id": 853, "name": "Mystery"}, {"id": 878, "name": "Science Fiction"}]	[{"cast_id": 2, "character": "Bruce Wayne / Batman", "..."}, {"cast_id": 3, "character": "Alfred Pennyworth", "..."}]	[{"credit_id": "52fe4781c3a36847f81398c3", "department": "Casting", "job": "Casting Director", "name": "Deborah A. Simon", "gender": "female", "slug": "deborah-a-simon"}]
4	49529	John Carter	John Carter is a war-weary, former military ca...	[{"id": 28, "name": "Action"}, {"id": 12, "name": "Adventure"}, {"id": 14, "name": "Fantasy"}, {"id": 878, "name": "Science Fiction"}]	[{"id": 818, "name": "based on novel"}, {"id": 853, "name": "Mystery"}, {"id": 878, "name": "Science Fiction"}]	[{"cast_id": 5, "character": "John Carter", "..."}, {"cast_id": 6, "character": "Deena", "..."}]	[{"credit_id": "52fe479ac3a36847f81398c3", "department": "Casting", "job": "Casting Director", "name": "Deborah A. Simon", "gender": "female", "slug": "deborah-a-simon"}]

```
In [9]: #missing data
movies.isnull().sum()

Out[9]: movie_id    0
title            0
overview         3
genres           0
keywords         0
cast             0
crew             0
dtype: int64

In [10]: #dropping the missing data(in overview)
movies.dropna(inplace=True)

In [11]: #missing data
movies.isnull().sum()

Out[11]: movie_id    0
title            0
overview         0
genres           0
keywords         0
```



```
In [12]: #duplicated data
movies.duplicated().sum()

Out[12]: 0

In [13]: movies.iloc[0].genres

Out[13]: '{"id": 28, "name": "Action"}, {"id": 12, "name": "Adventure"}, {"id": 14, "name": "Fantasy"}, {"id": 878, "name": "Science Fiction"}'
```

## convert string data to list

```
In [13]: movies.iloc[0].genres

Out[13]: '{"id": 28, "name": "Action"}, {"id": 12, "name": "Adventure"}, {"id": 14, "name": "Fantasy"}, {"id": 878, "name": "Science Fiction"}'

In [14]: # '{"id": 28, "name": "Action"}, {"id": 12, "name": "Adventure"}, {"id": 14, "name": "Fantasy"}, {"id": 878, "name": "Science Fiction"}'
# we need to convert the above format to the below format
# ['Action', 'Adventure', 'Fantasy', 'SciFi'] --preprocessing

In [15]: def convert(obj):
    L = []
    for i in ast.literal_eval(obj):
        L.append(i['name'])
    return L

In [16]: import ast
ast.literal_eval('{"id": 28, "name": "Action"}, {"id": 12, "name": "Adventure"}, {"id": 14, "name": "Fantasy"}, {"id": 878, "name": "Science Fiction"}')

Out[16]: ['id': 28, 'name': 'Action'],
{'id': 12, 'name': 'Adventure'},
{'id': 14, 'name': 'Fantasy'},
{'id': 878, 'name': 'Science Fiction']
```

```
In [17]: movies['genres'].apply(convert)

Out[17]: 0      [Action, Adventure, Fantasy, Science Fiction]
1      [Adventure, Fantasy, Action]
2      [Action, Adventure, Crime]
3      [Action, Crime, Drama, Thriller]
4      [Action, Adventure, Science Fiction]
...
4804     [Action, Crime, Thriller]
4805     [Comedy, Romance]
4806     [Comedy, Drama, Romance, TV Movie]
4807     []
4808     [Documentary]
Name: genres, Length: 4806, dtype: object
```

```
In [18]: movies['genres'] = movies['genres'].apply(convert)
```

```
In [19]: movies.head()
```

```
Out[19]:
```

	movie_id	title	overview	genres	keywords	cast	crew
0	19995	Avatar	In the 22nd century, a paraplegic Marine is di...	[Action, Adventure, Fantasy, Science Fiction]	[{"id": 1463, "name": "culture clash"}, {"id": ...	[{"cast_id": 242, "character": "Jake Sully", "...	[{"credit_id": "52fe48009251416c750aca23", "de...
1	285	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	[Adventure, Fantasy, Action]	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	[{"cast_id": 4, "character": "Captain Jack Spa...	[{"credit_id": "52fe4232c3a36847f800b579", "de...
2	206647	Spartan	A cryptic message from Rome's past sends him	[Action, Adventure,	[{"id": 470, "name": "enu/L /id": 818	[{"cast_id": 1, "character": "Tamec	[{"credit_id":

## convert data using AST to abstract grammer

```
In [27]: def fetch_director(obj):
L = []
for i in ast.literal_eval(obj):
    if i['job'] == 'Director':
        L.append(i['name'])
        break
return L
```

```
In [28]: movies['crew'].apply(fetch_director)

Out[28]: 0      [James Cameron]
1      [Gore Verbinski]
2      [Sam Mendes]
3      [Christopher Nolan]
4      [Andrew Stanton]
...
4804   [Robert Rodriguez]
4805   [Edward Burns]
4806   [Scott Smith]
4807   [Daniel Hsia]
4808   [Brian Herzlinger]
Name: crew, Length: 4806, dtype: object
```

```
In [29]: movies['crew'] = movies['crew'].apply(fetch_director)
```

```
In [30]: movies.head()
```

```
Out[30]:
```

	movie_id	title	overview	genres	keywords	cast	crew
0	19995	Avatar	In the 22nd century, a paraplegic Marine is di...	[Action, Adventure, Fantasy, Science Fiction]	[culture clash, future, space war, space colon...	[Sam Worthington, Zoe Saldana, Sigourney Weaver]	[James Cameron]
1	285	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	[Adventure, Fantasy, Action]	[ocean, drug abuse, exotic island, east india ...	[Johnny Depp, Orlando Bloom, Keira Knightley]	[Gore Verbinski]
2	206647	Spectre	A cryptic message from Bond's past sends him o...	[Action, Adventure, Crime]	[spy, based on novel, secret agent, sequel, mi...	[Daniel Craig, Christoph Waltz, Léa Seydoux]	[Sam Mendes]
3	49026	The Dark Knight Rises	Following the death of District Attorney Harve...	[Action, Crime, Drama, Thriller]	[dc comics, crime fighter, terrorist, secret i...	[Christian Bale, Michael Caine, Gary Oldman]	[Christopher Nolan]
4	49529	John Carter	John Carter is a war-weary, former military ca...	[Action, Adventure, Science Fiction]	[based on novel, mars, medallion, space travel...	[Taylor Kitsch, Lynn Collins, Samantha Morton]	[Andrew Stanton]

```
In [31]: movies['overview'][0]

Out[31]: 'In the 22nd century, a paraplegic Marine is dispatched to the moon Pandora on a unique mission, but becomes torn between following orders and protecting an alien civilization.'
```

```
In [32]: movies['overview'] = movies['overview'].apply(lambda x:x.split())
```

```
In [33]: movies.head()
```

```
Out[33]:
```

	movie_id	title	overview	genres	keywords	cast	crew
0	19995	Avatar	[In, the, 22nd, century,, a, paraplegic, Marin...	[Action, Adventure, Fantasy, Science Fiction]	[culture clash, future, space war, space colon...	[Sam Worthington, Zoe Saldana, Sigourney Weaver]	[James Cameron]
1	285	Pirates of the Caribbean: At World's End	[Captain, Barbossa,, long, believed, to, be, d...	[Adventure, Fantasy, Action]	[ocean, drug abuse, exotic island, east india ...	[Johnny Depp, Orlando Bloom, Keira Knightley]	[Gore Verbinski]
2	206647	Spectre	[A, cryptic, message, from, Bond's, past, send...	[Action, Adventure, Crime]	[spy, based on novel, secret agent, sequel, mi...	[Daniel Craig, Christoph Waltz, Léa Seydoux]	[Sam Mendes]
3	49026	The Dark Knight Rises	[Following, the, death, of, District, Attorney...	[Action, Crime, Drama, Thriller]	[dc comics, crime fighter, terrorist, secret i...	[Christian Bale, Michael Caine, Gary Oldman]	[Christopher Nolan]
4	49529	John Carter	[John, Carter, is, a, war-weary,, former, mili...	[Action, Adventure, Science Fiction]	[based on novel, mars, medallion, space travel...	[Taylor Kitsch, Lynn Collins, Samantha Morton]	[Andrew Stanton]

```
In [34]: movies['genres'].apply(lambda x:[i.replace(" ", "") for i in x])

Out[34]: 0      [Action, Adventure, Fantasy, ScienceFiction]
1      [Adventure, Fantasy, Action]
2      [Action, Adventure, Crime]
3      [Action, Crime, Drama, Thriller]
4      [Action, Adventure, ScienceFiction]
...
4804      [Action, Crime, Thriller]
4805      [Comedy, Romance]
4806      [Comedy, Drama, Romance, TVMovie]
4807      []
4808      [Documentary]
```

## Converting list data into tags

```
In [37]: movies['tags'] = movies['overview'] + movies['genres'] + movies['keywords'] + movies['cast'] + movies['crew']
```

```
In [38]: movies.head()
```

```
Out[38]:
```

	movie_id	title	overview	genres	keywords	cast	crew	tags
0	19995	Avatar	[In, the, 22nd, century,, a, paraplegic, Marin...	[Action, Adventure, Fantasy, ScienceFiction]	[cultureclash, future, spacewar, spacecolony, ...	[SamWorthington, ZoeSaldana, SigourneyWeaver]	[JamesCameron]	[In, the, 22nd, century,, a, paraplegic, Marin...
1	285	Pirates of the Caribbean: At World's End	[Captain, Barbossa,, long, believed, to, be, d...	[Adventure, Fantasy, Action]	[ocean, drugabuse, exoticioland, eastindiatrad...	[JohnnyDepp, OrlandoBloom, KeiraKnightley]	[GoreVerbinski]	[Captain, Barbossa,, long, believed, to, be, d...
2	206647	Spectre	[A, cryptic, message, from, Bond's, past, send...	[Action, Adventure, Crime]	[spy, basedonnovel, secretagent, sequel, mi6, ...	[DanielCraig, ChristophWaltz, LeaSeydoux]	[SamMendes]	[A, cryptic, message, from, Bond's, past, send...
3	49026	The Dark Knight Rises	[Following, the, death, of, District, Attorney...	[Action, Crime, Drama, Thriller]	[dcomics, crimefighter, terrorist, secretiden...	[ChristianBale, MichaelCaine, GaryOldman]	[ChristopherNolan]	[Following, the, death, of, District, Attorney...
4	49529	John Carter	[John, Carter, is, a, war-weary,, former, mili...	[Action, Adventure, ScienceFiction]	[basedonnovel, mars, medallion, spacetravel, p...	[TaylorKitsch, LynnCollins, SamanthaMorton]	[AndrewStanton]	[John, Carter, is, a, war-weary,, former, mili...

```
In [39]: new_df = movies[['movie_id','title','tags']]
```

```
Out[41]:
```

	movie_id	title	tags
0	19995	Avatar	In the 22nd century, a paraplegic Marine is di...
1	285	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...
2	206647	Spectre	A cryptic message from Bond's past sends him o...
3	49026	The Dark Knight Rises	Following the death of District Attorney Harve...
4	49529	John Carter	John Carter is a war-weary, former military ca...

```
In [43]: import nltk
from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()
```

```
In [44]: def stem(text):
y = []
for i in text.split():
y.append(ps.stem(i))

return " ".join(y)
```

```
In [45]: new_df['tags'] = new_df['tags'].apply(stem)
```

## Vectorization and recommendations

```
In [49]: from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer(max_features=5000,stop_words='english')
```

```
In [50]: vectors = cv.fit_transform(new_df['tags']).toarray()
```

```
In [51]: vectors
```

```
Out[51]: array([[0, 0, 0, ..., 0, 0, 0],
[0, 0, 0, ..., 0, 0, 0],
[0, 0, 0, ..., 0, 0, 0],
...,
[0, 0, 0, ..., 0, 0, 0],
[0, 0, 0, ..., 0, 0, 0],
[0, 0, 0, ..., 0, 0, 0]])
```

```
In [52]: vectors[0]
```

```
Out[52]: array([0, 0, 0, ..., 0, 0, 0])
```

```
In [53]: cv.get_feature_names()
```

```
Out[53]: ['000',
'007',
'110']
```

```

In [54]: ps.stem('dancer')
Out[54]: 'dancer'

In [55]: ps.stem('in the 22nd century, a paraplegic marine is dispatched to the moon pandora on a unique mission, but becomes
Out[55]: 'in the 22nd century, a paraplegic marine is dispatched to the moon pandora on a unique mission, but becomes torn be
tween following orders and protecting an alien civilization. action adventure fantasy sciencefiction cultureclash fu
ture spacewar spacecolony society spacetravel futuristic romance space alien tribe alienplanet cgi marine soldier ba
ttle loveaffair antiwar powerrelations mindandsoul 3d samworthington zoesaldana sigourneyweaver jamescameron'

In [56]: from sklearn.metrics.pairwise import cosine_similarity

In [57]: similarity = cosine_similarity(vectors)

In [58]: sorted(list(enumerate(similarity[0])),reverse=True,key=lambda x:x[1])[1:6]
Out[58]: [(1216, 0.28676966733820225),
(2409, 0.26901379342448517),
(3730, 0.2605130246476754),
(507, 0.255608593705383),
(539, 0.25038669783359574)]

In [59]: def recommend(movie):
    movie_index = new_df[new_df['title'] == movie].index[0]
    distances = similarity[movie_index]
    movies_list = sorted(list(enumerate(distances)),reverse=True,key=lambda x:x[1])[1:6]

    for i in movies_list:
        print(new_df.iloc[i[0]].title)
        #print(i)

In [60]: recommend('Batman Begins')

The Dark Knight
Batman
Batman
The Dark Knight Rises
10th & Wolf

In [61]: new_df.iloc[1216].title
Out[61]: 'Aliens vs Predator: Requiem'

In [62]: import pickle

In [63]: pickle.dump(new_df,open('movies.pkl','wb'))

In [65]: pickle.dump(new_df.to_dict(),open('movie_dict.pkl','wb'))

In [66]: pickle.dump(similarity,open('similarity.pkl','wb'))

In [ ]:

```

## Streamlit

Streamlit is a free and open-source framework to rapidly build and share beautiful machine learning and data science web apps. It is a Python-based library specifically designed for machine learning engineers.

The screenshot shows a PyCharm IDE with a project named 'movie-recommender-system'. The file explorer on the left shows a directory structure with files like 'app.py', 'movie\_dict.pkl', 'movies.pkl', and 'similarity.pkl'. The main editor displays the 'app.py' file with the following code:

```
1 import streamlit as st
2 import pickle
3 import pandas as pd
4 import requests
5
6 def fetch_poster(movie_id):
7     response = requests.get('http://api.themoviedb.org/3/movie/{}?api_key=8265bd1679663a7e11c296584c2c6e2d')
8     data = response.json()
9     return "https://image.tmdb.org/t/p/w500/" + data['poster_path']
10
11 def recommend(movie):
12     movie_index = movies[movies['title'] == movie].index[0]
13     distances = similarity[movie_index]
14     movies_list = sorted(list(enumerate(distances)), reverse=True, key=lambda x: x[1])[1:6]
15
16 fetch_poster()
```

The terminal at the bottom shows the command 'streamlit run app.py' being executed, and the output indicating that the app is running on 'http://localhost:8501'.

## Output

### Movie Recommender System

How would you like to be contacted?

John Carter

Recommend

### Movie Recommender System

How would you like to be contacted?

Spectre

Avatar

Pirates of the Caribbean: At World's End

Spectre

The Dark Knight Rises

John Carter

Spider-Man 3

Tangled

Avengers: Age of Ultron


## Movie Recommender System

How would you like to be contacted?

Spectre

Recommend

Quantum of Solace Skyfall Never Say Never From Russia with Love Octopussy



Please replace st\_beta\_columns with st\_columns.

## 5.2 Data set

### Files Dataset

We gather a data set of movies from different sources. These datasets include all the details of the movies released over various years. The following are the source links from where we gather all data:

- IMDB5000MovieDataset: <https://www.kaggle.com/carolzhangdc/imdb-5000movie-dataset>

- The Movies Dataset: <https://www.kaggle.com/rounakbanik/the-movies-dataset>

- List of movies in 2018:

[https://en.wikipedia.org/wiki/List\\_of\\_American\\_films\\_of\\_2018](https://en.wikipedia.org/wiki/List_of_American_films_of_2018)

- List of movies in 2019:

[https://en.wikipedia.org/wiki/List\\_of\\_American\\_films\\_of\\_2019](https://en.wikipedia.org/wiki/List_of_American_films_of_2019)

- List of movies in 2020:

[https://en.wikipedia.org/wiki/List\\_of\\_American\\_films\\_of\\_2020](https://en.wikipedia.org/wiki/List_of_American_films_of_2020)

## 5.3 Tools used

### API

We used TMDB API to access data including cast info, movie ratings, genres etc.

TMDB Movie API : <https://www.themoviedb.org/language=en-US>

Python programming language was used as the server-side programming language.

OS is used for file handling

NumPy and Pandas were used for array processing and data framing respectively

NLTK,

streamlit,

Countvectorizer,

sklearn,cosine\_similarity,Pickle etc...

## **6.CONCLUSION AND FUTURE SCOPE**

### **6.1 Conclusion**

Recommendation systems have become an important part of everyone's lives. With the enormous number of movies releasing worldwide every year, people often miss out on some amazing work of arts due to the lack of correct suggestion. Putting machine learning based Recommendation systems into work is thus very important to get the right recommendations. We saw content-based recommendation systems that although may not seem very effective on its own, but when combined with collaborative techniques can solve the cold start problems that collaborative filtering methods face when run independently. Similarly such systems can be improved further by applying neural network embedding to uplift the quality of recommendations and make them more user personalized.

Thus we conclude that studying various approaches towards recommendation engine is vital to come up with a collaborative engine that overcomes the shortcomings of these independent approaches and multiplies their benefits. Where independent approaches towards a movie recommendation system may have shortcomings, when combined the right way they will help users get the accurate recommendations for movies.

### **6.2. Future Recommendation**

Recommender system has developed for many years, which ever entered a low point. In the past few years, the development of machine learning, large-scale network and high performance computing is promoting new development in this field. We will consider the following aspects in future work.

**Introduce more precise and proper features of movie.**

Typical collaborative filtering recommendation use the rating instead of object features.

In the future we should extract features such as color and subtitle from movie which can provide a more accurate description for movie.

**Introduce user dislike movie list.**

The user data is always useful in recommender systems. In the future we will collect more user data and add user dislike movie list. We will input dislike movie list into the recommender system as well an generatescores that will be added to previous result. By this way we can improve the result of recommender system.

### **Introduce machine learning.**

For future study, dynamic parameters will be introduced into recommender system, we will use machine learning to adjust the weight of each feature automatically and find the most suitable weights.

### **Make the recommender system as an internal service.**

In the future, the recommender system is no longer an external website that will be just for testing. We will make it as an internal APIs for developers to invoke. Some movie lists in the website will be sorted by recommendation.

## **6.3 References**

[www.themoviedb.org/documentation/api](http://www.themoviedb.org/documentation/api)

Code Heroku “Movie Recommendation Engine | Machine Learning Projects”

[www.machinelearningplus.com/nlp/cosine-similarity](http://www.machinelearningplus.com/nlp/cosine-similarity)

Shani, G. & Gunawardana, A. (2011). "Evaluating recommendation systems", in Recommender systems handbook, ed: Springer, 2011, pp. 257-297.

Pazzani, M. J. & Billsus, D. (2007). "Content-based recommendation systems", in The adaptive web, ed: Springer, 2007, pp. 325-341.

Ricci F., Rokach L. & Shapira B. (2011). “Introduction to recommender system handbook”: Springer