# Movie Recommender Systems

A thesis submitted in partial fulfillment of the requirements for the award of the degree of

**B.Tech**

**in**

**Computer Science and Engineering**

By

**Sayak Nath (Roll No. 11200118036)**

**Bishwajit Prasad Gond (Roll No. 11200118054)**

**Atanu Paul (Roll No. 11200118059)**

**Abhishek Ghosh (Roll No. 11200118063)**

**Abhinaba Chakraborty (Roll No. 11200118064)**


**COMPUTER SCIENCE AND ENGINEERING**

**GOVERNMENT COLLEGE OF ENGINEERING AND LEATHER TECHNOLOGY**

**SALTLAKE SECTOR III – 700096**

**JAN 2022**

*In memory of*

*Dr. A. P. J. Abdul Kalam "Missile Man"*

*Former President of India*

# BONAFIDE CERTIFICATE

This is to certify that the project titled **Movie Recommender Systems** is a bonafide record of the work done by

**Sayak Nath (Roll No. 11200118036)**

**Bishwajit Prasad Gond (Roll No. 11200118054)**

**Atanu Paul (Roll No. 11200118059)**

**Abhishek Ghosh (Roll No. 11200118063)**

**Abhinaba Chakraborty (Roll No. 11200118064)**

in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering** of the **GOVERNMENT COLLEGE OF ENGINEERING AND TECHNOLOGY, KOLKATA**, during the year 2021-22.

**PROF. DEBAYAN GANGULY**
Guide

**DR. SANTANU HALDER**
Head of the Department

Project Viva-voce held on _____

**Internal Examiner**

**External Examiner**

# ABSTRACT

Recommender systems use algorithms to provide users with product or service recommendations. Recently, these systems have been using machine learning algorithms from the field of artificial intelligence. However, choosing a suitable machine learning algorithm for a Recommender system is difficult because of the number of algorithms described in the literature. Researchers and practitioners developing Recommender systems are left with little facts approximately the cutting-edge processes in algorithm usage. Moreover, the development of a Recommender system using a machine learning algorithm often has problems and open questions that must be evaluated, so software engineers know where to focus research efforts. This paper presents a systematic review of the literature that analyzes the use of machine learning algorithms in Recommender systems and identifies research opportunities for software engineering research. The study concludes that Bayesian and decision tree algorithms are widely used in Recommender systems because of their relative simplicity, and that requirement and design phases of Recommender system development appear to offer opportunities for further research.

*Keywords* : Recommender System, Machine Learning, Systematic Review

# ACKNOWLEDGEMENTS

We would like to express our deepest gratitude to the following people for guiding us through this course and without whom this project and the results achieved from it would not have reached completion.

**PROF. DEBAYAN GANGULY**, Assistant Professor, Department of Computer Science and Engineering, for helping us and guiding us in the course of this project. Without his guidance, we would not have been able to successfully complete this project. His patience and genial attitude is and always will be a source of inspiration to us.

**DR. SANTANU HALDER**, the Head of the Department, Department of Computer Science and Engineering, for allowing us to avail the facilities at the department.

We are also thankful to the faculty and staff members of the Department of Computer Science and Engineering, our individual parents and our friends for their constant support and help.

# TABLE OF CONTENTS

# LIST OF FIGURES

# CHAPTER 1

# INTRODUCTION

## 1.1 RECOMMENDER SYSTEM

Recommender systems (RS) are used to help users find new items or services, such as books, music, transportation or even people, based on information about the user, or the recommended item [1]. These systems also play an important role in decision-making, helping users to maximize profits [2] or minimize risks. Today, RSs are used in many information-based companies such as Google [3], Twitter [4], LinkedIn [5], and Netflix [6]. The field of RS has its origins in the mid-1990s with the introduction of Tapestry [7], the first RS. As the RS field evolved, researchers studied the use of algorithms from machine learning (ML), an area of artificial intelligence (AI). Machine learning has been studied since the late 1950s [8], with the emergence of the field of AI.Today, there is a plethora of ML algorithms (k-nearest neighbor [9], clustering [10], Bayes network [11], to name a few types), which are used in applications that range from vacuum cleaner robots [12] and assistance for disabled people to pattern recognition in images , or self-driving vehicles . The potential application of ML algorithms is vast and the field looks very promising. As stated earlier, ML algorithms are used in RS to give users higher suggestions. but, the ML discipline does now not have a clear separation system for its algorithms, in particular due to the number of methods and variations proposed in the literature. As a end result, it turns into tough and puzzling to pick an ML set of rules that fits someone's want when building an RS. similarly, researchers may discover it difficult to music the usage of ML algorithms in RS. software program Engineering (SE) is a subject that specializes in the introduction, development and protection of software with tools for programming, layout, and alertness of pc technological know-how information. The RS and ML fields can benefit from making use of SE terms and definitions. within the case of RSs the use of ML algorithms, researchers won't understand which SE web sites should have the greatest impact. another way to assist researchers and team of workers in determining which ML set of rules to use in RS, is to identify SE regions that may be stepped forward in RS improvement, RS and ML area research. studies on RS containing ML algorithms utilized in literature can help become aware of tendencies and offer steering for destiny research. For these motives, it has been determined to behavior systematic critiques to investigate a way to use actual RSs with ML

algorithms; what ML set of rules they use; and how the SE field can contribute to their development. it's far was hoping that, through this systematic evaluation, researchers and personnel can advantage extra understanding about the RS sector, and make more informed implementation or research choices.

### 1.1.1 TYPES OF RECOMMENDER SYSTEM

Through the years, recommender systems had been studied widely and are divided into different categories in line with the method being used. the types are collaborative fltering (CF), content based and context primarily based.

#### Collaboration filtering

Collaborative filtering (CF) makes use of user-generated numerical updates and is based totally in most cases on user history records available in the system [13], [14]. to be had historic facts facilitates to create a person profile and available information about the item is used to create an item profile. both the consumer profile and the item profile are used to create a recommendation plan. The Netflix competition has given splendid popularity to collaborative filtering, [14]. Collaborative screening is taken into consideration a basic and clean manner to get recommendations and make predictions approximately product sales. It has a few disadvantages which have caused the development of new strategies and techniques.

#### Content Based Recommender System

Content based [15] systems focus on the features of the products and aim at creating a user profile depending on the previous reviews and also a profile of the item in accordance with the features it provides and the reviews it has received [16], [14].It is observed that reviews usually contain product feature and user opinion in pairs [14], [15], [17]. It is observed that users' reviews contain a feature of the product followed by his/her opinion about the product. Content based recommendation systems help overcome sparsity problem that is faced in collaborative filtering based recommendation system.

#### Context Based Recommender System

Extending the user/item convention to the circumstances of the user to incorporate the contextual information is what is achieved in context-based recommender

systems [18]. This helps to abandon the cumbersome process of making the user fill a huge number of personal details.

## 1.2 CONTENT-BASED RECOMMENDER SYSTEM

It is a technique where individual user profiles are taken into account. It enhances the user's interest and predicts whether the user would be interested in eating at any particular restaurant or interested in seeing any particular movie [19]. It is also known as adaptive Filtering as it provides suggestions according to user's field of interest and adapts user's likes and dislikes. It represents the comparison between the content contained in the item with the content of items of user's interest. By using Bayesian hierarchical model, better user profiles for upcoming users is made by collecting feedbacks from the old users [20]. Content based collaborative filtering is more widely used to compare pure CF and pure Content-base. In CF the problem of sparsity is overcome (converting sparse user filled matrix into full user rating matrix) by using content-based prediction. Fig.1 displays the flow of information in a content based recommendation system. Relevant entities of an item and relations are kept together as input. Main features of items are extracted from item ontology. Features of items, user's ratings and user modeling data are applied to content based recommender system. After applying, various recommended items are given as output.

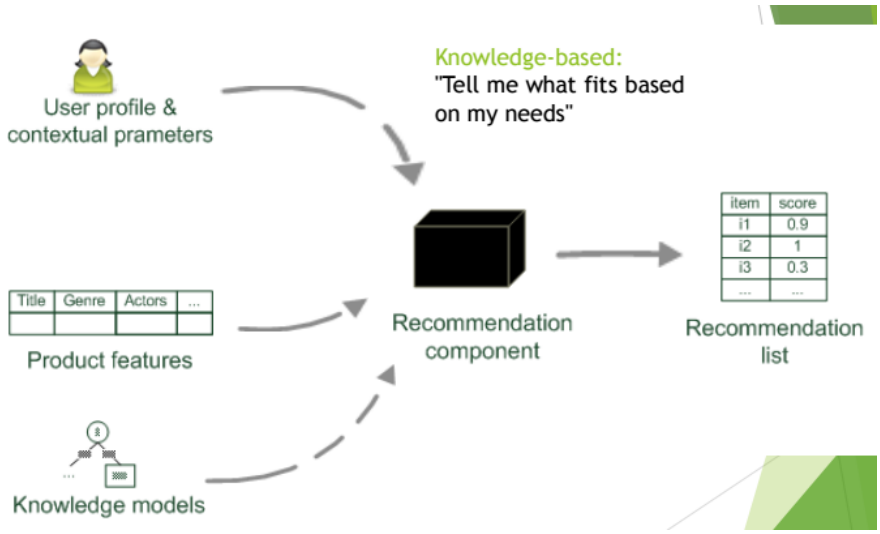


Figure 1.1: Content-Based Recommender System[1]

Different content based methods are using sentimental analysis for recommendation system. These methods are based on a variety of different models. Different researchers have studied the relevance of various techniques that can be implemented in content based recommendation systems. Accuracy and relevance of the

recommendation systems have become better by extensive research. Content-Boosted Collaborative filtering technique was implemented by Prem Melville et. Al. Recommendation system using content based technique and Bayesian Hierarchical Model (BHM) was built by Marko Balabanovic.

### 1.2.1 MOVIE RECOMMENDER SYSTEM

Here is an image of Movie Recommender system.

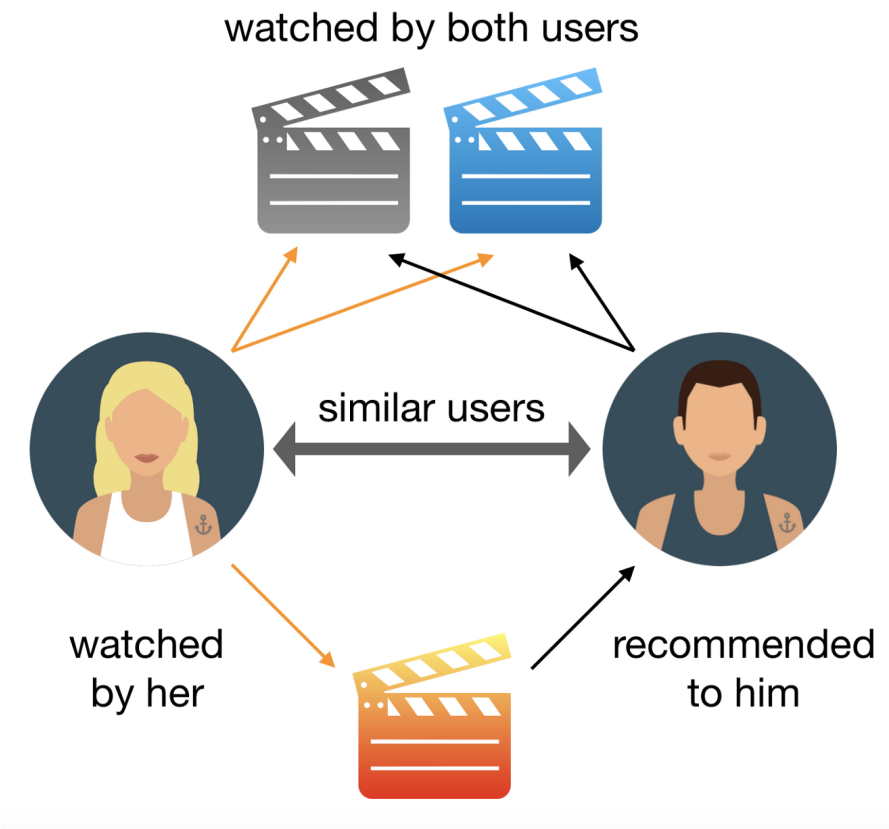


Figure 1.2: Movie Recommender system[]

# CHAPTER 2

# METHODOLOGY

## 2.1 DATA PREPROCESSING

Data preprocessing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model. When creating a machine learning project, it is not always a case that we come across the clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put in a formatted way. So for this, we use data preprocessing task. A real-world data generally contains noises, missing values, and maybe in an unusable format which cannot be directly used for machine learning models. Data preprocessing is required tasks for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model. It involves below steps:

1. Getting the dataset
2. Importing libraries
3. Importing datasets
4. Finding Missing Data
5. Encoding Categorical Data
6. Splitting dataset into training and test set
7. Feature scaling

### 2.1.1 Get the Dataset

To create a machine learning model, the first thing we required is a dataset as a machine learning model completely works on data. The collected data for a particular problem in a proper format is known as the dataset.

Dataset may be of different formats for different purposes, such as, if we want to create a machine learning model for business purpose, then dataset will be different with the dataset required for a liver patient. So each dataset is different from another dataset. To use the dataset in our code, we usually put it into a CSV file. However, sometimes, we may also need to use an HTML or xlsx file.

### 2.1.2 What is a CSV File?

CSV stands for ”Comma-Separated Values” files; it is a file format which allows us to save the tabular data, such as spreadsheets. It is useful for huge datasets and can use these datasets in programs.

### 2.1.3 Importing Libraries

In order to perform data preprocessing using Python, we need to import some predefined Python libraries. These libraries are used to perform some specific jobs. There are three specific libraries that we will use for data preprocessing, which are:

#### Numpy

Numpy Python library is used for including any type of mathematical operation in the code. It is the fundamental package for scientific calculation in Python. It also supports to add large, multidimensional arrays and matrices. So, in Python, we can import it as:

```
1 import numpy as np
```

#### Pandas

The last library is the Pandas library, which is one of the most famous Python libraries and used for importing and managing the datasets. It is an open-source data manipulation and analysis library. It will be imported as below:

```
1 import pandas as pd
```

## 2.2 VECTORIZATION

We know that most of the application has to deal with a large number of datasets. Hence, a non-computationally-optimal function can become a huge bottleneck in your algorithm and can take result in a model that takes ages to run. To make sure that the code is computationally efficient, we will use vectorization. Time complexity in the execution of any algorithm is very crucial deciding whether an application is reliable or not. To run a large algorithm in as much as optimal time possible is very important when it comes to real-time application of output. To do so, Python has some standard mathematical functions for fast operations on entire arrays of data without having to write loops. One of such library which contains such function is numpy. Let’s see how can we use this standard function in case of vectorization.

### 2.2.1 What is Vectorization?

Vectorization is used to speed up the Python code without using loop. Using such a function can help in minimizing the running time of code efficiently. Various operations are being performed over vector such as dot product of vectors which is also known as scalar product as it produces single output, outer products which results in square matrix of dimension equal to length X length of the vectors, Element wise multiplication which products the element of same indexes and dimension of the matrix remain unchanged.

### 2.2.2 Why Vectorization is important in Machine Learning?

Just like in the real-world we are interested in solving any kind of problem efficiently in such a way that the amount of error is reduced as much as possible. In machine learning, there's a concept of an optimization algorithm that tries to reduce the error and computes to get the best parameters for the machine learning model. So by using a vectorized implementation in an optimization algorithm we can make the process of computation much faster compared to Unvectorized Implementation. A basic example of Optimization Algorithm: Gradient Descent

## 2.3 COSINE SIMILARITY

In data analysis, Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space. It is defined to equal the cosine of the angle between them, which is also the same as the inner product of the same vectors normalized to both have length 1. These bounds apply for any number of dimensions, and the cosine similarity is most commonly used in high-dimensional positive spaces. For example, in information retrieval and text mining, each term is notionally assigned a different dimension and a document is characterised by a vector where the value in each dimension corresponds to the number of times the term appears in the document. Cosine similarity then gives a useful measure of how similar two documents are likely to be in terms of their subject matter. The technique is also used to measure cohesion within clusters in the field of data mining. One advantage of cosine similarity is its low-complexity, especially for sparse vectors: only the non-zero dimensions need to be considered. Cosine similarity is a metric used to determine 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. In this context, the two vectors I am talking about are arrays containing the word counts of two documents. As a similarity metric, how does cosine similarity differ from the number of common words?

When plotted on a multi-dimensional space, where each dimension corresponds to a word in the document, the cosine similarity captures the orientation (the angle) of the documents and not the magnitude. If you want the magnitude, compute the Euclidean distance instead. The cosine similarity is advantageous because even if the two similar documents are far apart by the Euclidean distance because of the size (like, the word ‘cricket’ appeared 50 times in one document and 10 times in another) they could still have a smaller angle between them. Smaller the angle, higher the similarity.

# APPENDIX A

# CODE ATTACHMENTS

## A.1 MOVIE RECOMMENDER SYSTEM

```
"""Movie_Recomend.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1y_s9fl3UohxpWMz1dqE8kh16FfZreeGw
"""

!cp "/content/drive/MyDrive/tmdb_5000_credits.csv.zip" "/content"

from google.colab import drive
drive.mount('/content/drive')

!cp "/content/drive/MyDrive/tmdb_5000_movies.csv.zip" "/content"

!7z x "tmdb_5000_movies.csv.zip"

!7z x "tmdb_5000_credits.csv.zip"

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

movies = pd.read_csv('tmdb_5000_movies.csv')
credits = pd.read_csv('tmdb_5000_credits.csv')

movies.head(1)

movies.shape

credits.head(1)

movies = movies.merge(credits,on='title') #joining the two databases on the title column

movies.head(1)
# budget
# homepage
# id
# original_language
# original_title
# popularity
# production_comapny
# production_countries
```

```
# release-date

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

movies.head(1)

import ast

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

movies.dropna(inplace=True) # drops tuples with non applicable values
    present
                            #in any column

movies['genres'] = movies['genres'].apply(convert)
movies.head(1)

movies['keywords'] = movies['keywords'].apply(convert)
movies.head(1)

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

def convert3(text):
    L = []
    counter = 0
    for i in ast.literal_eval(text):
        if counter < 3:
            L.append(i['name'])
        counter+=1
    return L

movies['cast'] = movies['cast'].apply(convert3)
movies.head(1)

movies['cast'][0]

def fetch_director(text):
    L = []
    for i in ast.literal_eval(text):
        if i['job'] == 'Director':
            L.append(i['name'])
    return L

movies['crew'] = movies['crew'].apply(fetch_director)

movies.head(1)

movies.sample(5)
```

```
95
96  def collapse(L):
97      L1 = []
98      for i in L:
99          L1.append(i.replace(" ",""))
100     return L1
101
102 movies['cast'] = movies['cast'].apply(collapse)
103 movies['crew'] = movies['crew'].apply(collapse)
104 movies['genres'] = movies['genres'].apply(collapse)
105 movies['keywords'] = movies['keywords'].apply(collapse)
106
107 movies.head(1)
108
109 movies['overview'] = movies['overview'].apply(lambda x:x.split())
110
111 movies['tags'] = movies['overview'] + movies['genres'] + movies['
    keywords'] + movies['cast'] + movies['crew']
112
113 new = movies.drop(columns=['overview','genres','keywords','cast','
    crew'])
114 new.head(1)
115
116 new['tags'] = new['tags'].apply(lambda x: " ".join(x))
117 new.head()
118
119 from sklearn.feature_extraction.text import CountVectorizer
120 cv = CountVectorizer(max_features=5000,stop_words='english')
121
122 vector = cv.fit_transform(new['tags']).toarray()
123
124 vector.shape
125
126 from sklearn.metrics.pairwise import cosine_similarity
127
128 similarity = cosine_similarity(vector)
129
130 similarity
131
132 new[new['title'] == 'Batman'].index[0]
133
134 def recommend(movie):
135     index = new[new['title'] == movie].index[0]
136     distances = sorted(list(enumerate(similarity[index])),reverse=
        True,key = lambda x: x[1])
137     for i in distances[1:6]:
138         print(new.iloc[i[0]].title)
139
140 x=input("Enter Movie Name : ")
141
142 recommend(x)
```

## Import Libraries

[1]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
```

## Import required files

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

### Exploring the imported files

[3]:
```
movies.head(1)
```

[3]:

| | budget | genres | homepage | id | keywords | original_language | original_title | overview | popularity | production_companies | production_countries | release_date |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 237000000 | [{"id": 28, "name": "Action"}, {"id": 12, "nam... | http://www.avatarmovie.com/ | 19995 | [{"id": 1463, "name": "culture clash"}, {"id":... | en | Avatar | In the 22nd century, a paraplegic Marine is di... | 150.437577 | [{"name": "Ingenious Film Partners", "id": 289... | [{"iso_3166_1": "US", "name": "United States o... | 2009-12-10 |

[4]:
```
movies.shape
```

[4]:
```
(4803, 20)
```

[5]:
```
credits.head(1)
```

[5]:

| | movie_id | title | cast | crew |
|---|---|---|---|---|
| 0 | 19995 | Avatar | [{"cast_id": 242, "character": "Jake Sully", "... | [{"credit_id": "52fe48009251416c750aca23", "de... |

## Merge the two files into one

[6]:
```
movies = movies.merge(credits,on='title') #joining the two databases on the title column
```

[7]:
```
movies.head(1)
# budget
# homepage
# id
# original_Language
# original_title
# popularity
# production_comapny
# production_countries
# release-date
```

[7]:

| | budget | genres | homepage | id | keywords | original_language | original_title | overview | popularity | production_companies | ... | runtime | spoken_languages | s |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 237000000 | [{"id": 28, "name": "Action"}, {"id": 12, "nam... | http://www.avatarmovie.com/ | 19995 | [{"id": 1463, "name": "culture clash"}, {"id":... | en | Avatar | In the 22nd century, a paraplegic Marine is di... | 150.437577 | [{"name": "Ingenious Film Partners", "id": 289... | ... | 162.0 | [{"iso_639_1": "en", "name": "English"}, {"iso... | Rel |

1 rows × 23 columns

## Selecting the necessary columns from to be used for processing

[8]:
```
movies = movies[['movie_id','title','overview','genres','keywords','cast','crew']]
```

[9]:
```
movies.head(1)
```

[9]:

| | 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, "nam... | [{"id": 1463, "name": "culture clash"}, {"id":... | [{"cast_id": 242, "character": "Jake Sully", "... | [{"credit_id": "52fe48009251416c750aca23", "de... |

## Function to convert the "genres" and "keywords" column values to a more suitable format

[10]:
```
import ast

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

movies.dropna(inplace=True) # drops tuples with non applicable values present
                            #in any column

movies['genres'] = movies['genres'].apply(convert)
movies.head(1)
```

[10]:

| | 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... |

[11]:
```
movies['keywords'] = movies['keywords'].apply(convert)
movies.head(1)
```

[11]:

| | 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... | [{"cast_id": 242, "character": "Jake Sully", "... | [{"credit_id": "52fe48009251416c750aca23", "de... |

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

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

## Function to find the names of first three cast members

[13]:
```
def convert3(text):
    L = []
    counter = 0
    for i in ast.literal_eval(text):
        if counter < 3:
            L.append(i['name'])
        counter+=1
    return L
```

[14]:
```
movies['cast'] = movies['cast'].apply(convert3)
movies.head(1)
```

[14]:

| | 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] | [{"credit_id": "52fe48009251416c750aca23", "de... |

[15]:
```
movies['cast'][0]
```

[15]:
```
['Sam Worthington', 'Zoe Saldana', 'Sigourney Weaver']
```

## Function to find the name of the Director from the list of crew members

[16]:
```
def fetch_director(text):
    L = []
    for i in ast.literal_eval(text):
        if i['job'] == 'Director':
            L.append(i['name'])
    return L

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

[17]:
```
movies.head(1)
```

[17]:

| | 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] |

[18]:
```
movies.sample(5)
```

[18]:

| | movie_id | title | overview | genres | keywords | cast | crew |
|---|---|---|---|---|---|---|---|
| 500 | 584 | 2 Fast 2 Furious | It's a major double-cross when former police o... | [Action, Crime, Thriller] | [miami, car race, sports car, los angeles, car... | [Paul Walker, Tyrese Gibson, Eva Mendes] | [John Singleton] |
| 3038 | 9455 | The Corruptor | Danny is a young cop partnered with Nick, a se... | [Action, Crime, Mystery, Thriller] | [new york, life and death, gang war, police, c... | [Mark Wahlberg, Chow Yun-fat, Byron Mann] | [James Foley] |
| 178 | 44896 | Rango | When Rango, a lost family pet, accidentally wi... | [Animation, Comedy, Family, Western, Adventure] | [sheriff, nevada, pet, rango, chameleon, las v... | [Johnny Depp, Isla Fisher, Ned Beatty] | [Gore Verbinski] |
| 3755 | 39781 | The Kids Are All Right | Two women, Nic and Jules, brought a son and da... | [Comedy, Drama] | [lesbian relationship, lesbian, dinner, motorc... | [Julianne Moore, Annette Bening, Mark Ruffalo] | [Lisa Cholodenko] |
| 3877 | 103663 | The Hunt | A teacher lives a lonely life, all the while s... | [Drama] | [father son relationship, denmark, lie, pedoph... | [Mads Mikkelsen, Alexandra Rapaport, Thomas Bo... | [Thomas Vinterberg] |

## Function to remove in-between spaces from certain columns ¶

[19]:
```
def collapse(L):
    L1 = []
    for i in L:
        L1.append(i.replace(" ",""))
    return L1
```

[20]:
```
movies['cast'] = movies['cast'].apply(collapse)
movies['crew'] = movies['crew'].apply(collapse)
movies['genres'] = movies['genres'].apply(collapse)
movies['keywords'] = movies['keywords'].apply(collapse)
```

[21]:
```
movies.head(1)
```

[21]:

| | movie_id | title | overview | genres | keywords | cast | crew |
|---|---|---|---|---|---|---|---|
| 0 | 19995 | Avatar | In the 22nd century, a paraplegic Marine is di... | [Action, Adventure, Fantasy, ScienceFiction] | [cultureclash, future, spacewar, spacecolony, ... | [SamWorthington, ZoeSaldana, SigourneyWeaver] | [JamesCameron] |

## Splitting values in "overview" column into list of words

[22]:
```
movies['overview'] = movies['overview'].apply(lambda x:x.split())
movies['overview'].sample(5)
```

[22]:
```
1716    [An, estranged, family, of, former, child, pro...
1764    [Slevin, is, mistakenly, put, in, the, middle,...
3509    [Tells, the, seemingly, random, yet, vitally, ...
2465    [Hollywood,, 1927:, As, silent, movie, star, G...
2965    [In, a, world, connected, by, YouTube,, iTunes...
Name: overview, dtype: object
```

## Creating a "tag" column using specific existing columns

[23]:
```
movies['tags'] = movies['overview'] + movies['genres'] + movies['keywords'] + movies['cast'] + movies['crew']
movies['tags'].sample(5)
```

[23]:
```
4497    [A, look, at, human, origins, in, the, very, p...
3486    [A, young, man, awakens, from, a, four-year, c...
155     [After, Dick, Harper, loses, his, job, at, Glo...
3717    [Friends,, family,, and, lovers, struggle, to,...
1914    [John, and, his, girlfriend, have, vowed, to, ...
Name: tags, dtype: object
```

## Creating a new table using suitable columns

[24]:
```
new = movies.drop(columns=['overview','genres','keywords','cast','crew'])
new.head(1)
```

[24]:

| | movie_id | title | tags |
|---|---|---|---|
| 0 | 19995 | Avatar | [In, the, 22nd, century,, a, paraplegic, Marin... |

## Changing the "tags" column into paragraph

[25]:
```
new['tags'] = new['tags'].apply(lambda x: " ".join(x))
new.head()
```

[25]:

| | 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... |

## Creating the ML model

[26]:
```
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer(max_features=5000,stop_words='english')
```

[27]:
```
vector = cv.fit_transform(new['tags']).toarray()
```

[28]:
```
vector.shape
```

[28]:
```
(4806, 5000)
```

[29]:
```
from sklearn.metrics.pairwise import cosine_similarity
```

[30]:
```
similarity = cosine_similarity(vector)
```

[31]:
```
similarity
```

[31]:
```
array([[1.        , 0.0860309 , 0.05735393, ..., 0.0244558 , 0.0270369 ,
        0.        ],
       [0.0860309 , 1.        , 0.0625    , ..., 0.02665009, 0.        ,
        0.        ],
       [0.05735393, 0.0625    , 1.        , ..., 0.02665009, 0.        ,
        0.        ],
       ...,
       [0.0244558 , 0.02665009, 0.02665009, ..., 1.        , 0.07537784,
        0.0489116 ],
       [0.0270369 , 0.        , 0.        , ..., 0.07537784, 1.        ,
        0.05407381],
       [0.        , 0.        , 0.        , ..., 0.0489116 , 0.05407381,
        1.        ]])
```

[32]:
```
new[new['title'] == 'Batman'].index[0]
```

[32]:
```
1362
```

## Function to recommend movies in response to the user input

[33]:
```
def recommend(movie):
    index = new[new['title'] == movie].index[0]
    distances = sorted(list(enumerate(similarity[index])),reverse=True,key = lambda x: x[1])
    for i in distances[1:6]:
        print(new.iloc[i[0]].title)
```

[34]:
```
x=input("Enter Movie Name : ")
```

```
Enter Movie Name :  The Avengers
```

## Calling the "recommend" function

[35]:
```
recommend(x)
```

```
Avengers: Age of Ultron
Captain America: Civil War
Iron Man 3
Captain America: The First Avenger
Iron Man
```

# REFERENCES


[1] Gediminas Adomavicius and Alexander Tuzhilin. "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions". In: *IEEE transactions on knowledge and data engineering* 17.6 (2005), pp. 734–749.

[2] Long-Sheng Chen, Fei-Hao Hsu, Mu-Chen Chen, and Yuan-Chia Hsu. "Developing recommender systems with the consideration of product profitability for sellers". In: *Information Sciences* 178.4 (2008), pp. 1032–1048.

[3] Jiahui Liu, Peter Dolan, and Elin Rønby Pedersen. "Personalized news recommendation based on click behavior". In: *Proceedings of the 15th international conference on Intelligent user interfaces*. 2010, pp. 31–40.

[4] Amr Ahmed, Bhargav Kanagal, Sandeep Pandey, Vanja Josifovski, Lluis Garcia Pueyo, and Jeff Yuan. "Latent factor models with additive and hierarchically-smoothed user preferences". In: *Proceedings of the sixth ACM international conference on Web search and data mining*. 2013, pp. 385–394.

[5] Mario Rodriguez, Christian Posse, and Ethan Zhang. "Multiple objective optimization in recommender systems". In: *Proceedings of the sixth ACM conference on Recommender systems*. 2012, pp. 11–18.

[6] Harald Steck. "Evaluation of recommendations: rating-prediction and ranking". In: *Proceedings of the 7th ACM conference on Recommender systems*. 2013, pp. 213–220.

[7] David Goldberg, David Nichols, Brian M Oki, and Douglas Terry. "Using collaborative filtering to weave an information tapestry". In: *Communications of the ACM* 35.12 (1992), pp. 61–70.

[8] Henrik H Martens. "Two notes on machine Learning". In: *Information and Control* 2.4 (1959), pp. 364–379.

[9] Edward A Patrick and Frederic P Fischer III. "A generalized k-nearest neighbor rule". In: *Information and control* 16.2 (1970), pp. 128–152.

[10] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. "Data clustering: a review". In: *ACM computing surveys (CSUR)* 31.3 (1999), pp. 264–323.

[11] Nir Friedman, Dan Geiger, and Moises Goldszmidt. "Bayesian network classifiers". In: *Machine learning* 29.2 (1997), pp. 131–163.

[12] D Burhams and Michael Kandefer. "Dustbot: Bringing Vacuum-Cleaner Agent to Life". In: *Accessible Hands-on Artificial Intelligence and Robotics Education* (2004), pp. 22–24.

[13] Xiaoyuan Su and Taghi M Khoshgoftaar. “A survey of collaborative filtering techniques”. In: *Advances in artificial intelligence* 2009 (2009).

[14] Yongfeng Zhang. “Incorporating phrase-level sentiment analysis on textual reviews for personalized recommendation”. In: *Proceedings of the eighth ACM international conference on web search and data mining*. 2015, pp. 435–440.

[15] Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. “Content-based recommender systems: State of the art and trends”. In: *Recommender systems handbook* (2011), pp. 73–105.

[16] Ivan Titov and Ryan McDonald. “A joint model of text and aspect ratings for sentiment summarization”. In: *proceedings of ACL-08: HLT*. 2008, pp. 308–316.

[17] Xiaowen Ding, Bing Liu, and Philip S Yu. “A holistic lexicon-based approach to opinion mining”. In: *Proceedings of the 2008 international conference on web search and data mining*. 2008, pp. 231–240.

[18] Umberto Panniello, Alexander Tuzhilin, Michele Gorgoglione, Cosimo Palmisano, and Anto Pedone. “Experimental comparison of pre-vs. post-filtering approaches in context-aware recommender systems”. In: *Proceedings of the third ACM conference on Recommender systems*. 2009, pp. 265–268.

[19] Chumki Basu, Haym Hirsh, William Cohen, et al. “Recommendation as classification: Using social and content-based information in recommendation”. In: *Aaai/iaai*. 1998, pp. 714–720.

[20] Yi Zhang and Jonathan Koren. “Efficient bayesian hierarchical user modeling for recommendation system”. In: *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. 2007, pp. 47–54.