Movie Recommender System using Content Based Filtering

A PROJECT REPORT

Submitted by

ARYAN KASHYAP	20BAI10072
SHREY KHANDUJA	20BAI10194
SAGAR MAHESHWARI	20BAI10339
ALOK KHANSALI	20BAI10347

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING WITH SPECIALIZATION IN ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

VIT BHOPAL UNIVERSITY, KOTHRIKALAN, SEHORE

MADHYA PRADESH - 466114

DEC 2021

BONAFIDE CERTIFICATE

Certified that this project report titled **"Movie Recommender System Using Content Based Filtering"** is the bonafide work of **"Aryan Kashyap (20BAI10072) Shrey Khanduja (20BAI10194) Sagar Maheshwari (20BAI10339) Alok Khansali (20BAI10347)"** who carried out the project work under my supervision. Certified further that to the best of my knowledge the work reported at this time does not form part of any other project/research work based on which a degree or award was conferred on an earlier occasion on this or any other candidate.

PROGRAM CHAIR
Dr. S.Sountharrajan
Division Head - Artificial Intelligence
School of Computer Science and Engineering
VIT BHOPAL UNIVERSITY

PROJECT GUIDE
Dr. Elangovan G
Assistant Professor
School of Computer Science and Engineering
VIT BHOPAL UNIVERSITY

The Project Exhibition I Examination is held on 22 December 2021.

ACKNOWLEDGEMENT

First and foremost, I would like to thank the Lord Almighty for His presence and immense blessings throughout the project work.

I wish to express my heartfelt gratitude to Dr S. SOUNTHARRAJAN, Division Head - Artificial Intelligence, School of Computer Science and Engineering for much of his valuable support encouragement in carrying out this work.

I would like to thank my internal guide Dr G ELANGOVAN, for continually guiding and actively participating in our project, giving valuable suggestions to complete the project work.

I would like to thank all the technical and teaching staff of the School of School of Computer Science and Engineering, who extended directly or indirectly all support.

Last, but not least, I am deeply indebted to my parents who have been the greatest support while I worked day and night for the project to make it a success.

LIST OF ABBREVIATIONS

Sl. No.	Abbreviation	Full form
1	RS	Recommender System
2	IDE	Integrated Development Environment
3	SKLEARN	Sci-kit learn
4	OS	Operating System
5	COLLAB	Collaborative
6	NLTK	Natural Language Toolkit
7	AST	Abstract Syntax Trees

LIST OF FIGURES AND GRAPHS

LIST OF TABLES

ABSTRACT

Recommender System is a tool helping users find content and overcome information overload. It predicts interests of users and makes recommendation according to the interest model of users. The original content-based recommender system is the continuation and development of collaborative filtering, which doesn't need the user's evaluation for items. With the improvement of machine learning, current content-based recommender system can build profile for users and products respectively. Building or updating the profile according to the analysis of items that are bought or visited by users. The foundation of content-based algorithm is acquisition and quantitative analysis of the content. As the research of acquisition and filtering of text information are mature, many current content-based recommenders systems make recommendation according to the analysis of text information.

TABLE OF CONTENTS

CHAPTER 1

PROJECT DESCRIPTION AND OUTLINE

1.1 Introduction

In today's digital world where there is an endless variety of content to be consumed like books, videos, articles, movies, etc., finding the content of one's liking has become an irksome task. On the other hand, digital content providers want to engage as many users on their service as possible for the maximum time. This is where recommender system comes into picture where the content providers recommend users the content according to the users' liking. In this paper we have proposed a movie recommender system. The objective of this is to provide accurate movie recommendations to users.

1.2 Motivation for the work

Recommendation systems help users find and select items (e.g., books, movies, restaurants) from the huge number available on the web or in other electronic information sources. Given a large set of items and a description of the user's needs, they present to the user a small set of the items that are well suited to the description. Similarly, a movie recommendation system provides a level of comfort and personalization that helps the user interact better with the system and watch movies that cater to his needs. Providing this level of comfort to the user was our primary motivation in opting for movie recommendation system as our Project. The chief purpose of our system is to recommend movies to its users based on their viewing history and ratings that they provide. Personalized recommendation engines help millions of people narrow the universe of potential films to fit their unique tastes. Collaborative filtering and content-based filtering are the are prime approaches to provide recommendation to users. Both are best applicable in specific scenarios because of their respective ups and downs. In this paper we have proposed a content-based approach. A movie recommendation system provides a level of comfort and personalization that helps the user interact better with the system and watch movies that cater to his needs. Providing this level of comfort to the user was our primary motivation in opting for movie recommendation system as our BE Project.

1.3 Content-Based Movie Recommendation System Introduction

Recommendation systems help users find and select items (e.g., books, movies, restaurants) from the huge number available on the web or in other electronic information sources. Given a large set of items and a description of the user's needs, they present to the user a small set of the items that are well suited to the description. Personalized recommendation engines help millions of people narrow the universe of potential films to fit their unique tastes. Collaborative filtering and content-based filtering are the are prime approaches to provide recommendation to users. Both of them are best applicable in specific scenarios because of their respective ups and downs. In this paper we have proposed a mixed approach such that both the algorithms complement each other thereby improving performance and accuracy of the of our system. Once, we know the likings of the user we can embed him/her in an embedding space using the feature vector generated and recommend him/her according to his/her choice. During recommendation, the similarity metrics (We will talk about it later) are calculated from the item's feature vectors and the user's preferred feature vectors from his/her previous records. Then, the top few are recommended. Content-based filtering does not require other user's data during recommendations to one user.

1.4 Problem Statement

The rapid growth of data collection has led to a new era of information. We now live in what some call the "era of abundance". For any given product, there are sometimes thousands of options to choose from. Think of the examples above: streaming videos, social networking, online shopping; the list goes on. Recommendation Systems are a type of information filtering systems as they improve the quality of search results and provides items that are more relevant to the search item or are related to the search history of the user.

CHAPTER 2

RELATED WORK INVESTIGATION

2.1 INTRODUCTION

Learnt python programming language for the project, which includes basic's that are syntax of the language, declaration of variables, if-else conditional statements, for, while and do-while loops, function creation etc.

For the project creation, installed jupyter notebook/google collab, learned the libraries/module for the data analysis which are NumPy, pandas, scikit-learn, for visualizing the data applied these libraries – Matplotlib, Seaborn, Plotly and also cufflinks.

There are two types of recommender system – Content based, and Collaborative filtering and modern approach also includes both of them also called a hybrid approach.

2.2 Method – How do Content Based Recommender Systems work?

A content-based recommender works with data that the user provides, either explicitly (rating) or implicitly (clicking on a link). Based on that data, a user profile is generated, which is then used to make suggestions to the user. As the user provides more inputs or takes actions on the recommendations, the engine becomes more and more accurate.

2.3 Existing Approaches

2.3.1 Content-based filtering system:

Content-based filtering system is a type of recommender system that attempts to guess what a user may like based on that user's activity. Content-based filtering makes recommendations by using keywords and attributes assigned to objects in a database (e.g., items in an online marketplace) and matching them to a user profile.

2.3.2 Collaborative Filtering System:

Collaborative filtering system recommends items based on similarity measures between users and/or items. The system recommends items preferred by similar users. This is based on the scenario where a person asks his friends, who have similar tastes, to recommend him some movies.

2.4 Pros and Cons of stated approaches:

2.4.1 content-based filtering System:

Advantages of content-based filtering are:

- They capable of recommending unrated items
- We can easily explain the working of recommender system by listing the Content features of an item.
- Content-based recommender systems use need only the rating of the concerned user, and not any other user of the system.

Disadvantages of content-based filtering are:

- It does not work for a new user who has not rated any item yet as enough ratings are required content-based recommender evaluates the user preferences and provides accurate recommendations.
- No recommendation of serendipitous items.
- Limited Content Analysis- The recommender does not work if the system fails to distinguish the items that a user likes from the items that he does not like.

2.4.2 Collaborative Filtering System:

Advantages of collaborative filtering-based systems:

- It is dependent on the relation between users which implies that it is content-independent.
- CF recommender systems can suggest serendipitous items by observing similar-minded people's behavior.
- They can make real quality assessment of items by considering other peoples experience.

Disadvantages of collab filtering are:

- Early rater problem: Collab filtering systems cannot provide recommendations for new items since there are no user ratings on which to base a prediction.

- Gray sheep: For CF based system to work, group with similar characteristics are needed. Even if such groups exist, it will be very difficult to recommend users who do not consistently agree or disagree to these groups.
- Sparsity problem: In most cases, the number of items exceed the number of users by a great margin which makes it difficult to find items that are rated by enough people.

2.5 Issues/observations from investigation

- **Lack of Data:** Perhaps the biggest issue facing recommender systems is that they need a lot of data to effectively make recommendations. It's no coincidence that the companies most identified with having excellent recommendations are those with a lot of consumer user data: Google, Amazon, Netflix, Last.fm. The more item and user data a recommender system has to work with, the stronger the chances of getting good recommendations.
- **Changing Data:** Past behavior [of users] is not a good tool because the trends are always changing
- **Changing User Preferences**

2.6 Challenges faced in developing recommendation system

- **Cold Start:** This problem arises when new users or new items are added to the system, a new item can't recommend to users initially when it is introduced to the recommendation system without any rating or reviews and hence it is hard to predict the choice or interest of users which leads to less accurate recommendations.
- **Synonymy:** Synonymy arises when a single item is represented with two or more different names or listings of items having similar meanings, in such condition, the recommendation system can't recognize whether the terms show various items or the same item.
- **Scalability:** One biggest issue is the scalability of algorithms having real-world datasets under the recommendation system, a huge changing data is generated by user-item interactions in the form of ratings and reviews and consequently, scalability is a big concern for these datasets.

  Recommendation systems interpret results on large datasets inefficiently, some advanced large-scaled methods are required for this issue.

Conclusion:

For sure, the recommendation system is unturned to exist in the e-commerce businesses with the help of collaborative or content-based filtering to predict different items and yes, users are most satisfied with the products recommended to them. The recommending system provides new ways of finding or extracting personalized information on the internet. It enables users to have access to the products and services easily within limited periods of time. There are the most advanced methods in order to design the high-quality and fine-tuned recommendations engine such as machine learning, deep learning, neural networks, etc. It is well worth saying recommendation systems improve the usage of machine learning along with big data analytics.

CHAPTER 3
REQUIREMENT ARTIFACTS

3.1 Introduction

Recommender systems (RS) help users to choose or discover new content by monitoring and aggregating user's data, and displaying recommendations. This research area has its origins in the mid-1990s with the introduction of Tapestry, the first recommender system. With the evolution of technology and the popularization of the Internet, academia and industry adopted recommender systems for many application domains including, but not limited to, movies, books, tourism, social networks, academia (article recommendation), and also technical domains such as programming (code segment recommendation), and medicine (drug recommendations)

3.2 Hardware and Software requirements

Hardware:
CPU: 2 x 64-bit 2.8 GHz

Ram: Systems with 2GB RAM (4GB preferable)

Storage: 10gb (Preferred)

Software:
Operating system: Linux- Ubuntu 16.04 to 17.10, Windows 8 to 11 or MacOS

Browser: Google Chrome or Mozilla Firefox (latest version)

Environment: Anaconda Navigator with Python 3.6 (Latest Preferred)

Python Modules such as NumPy, Pandas, Scikit-learn & Nltk

3.3 Specific Project requirements

3.3.1 Data requirement

For the system, we'll use the open-source **TMDB 5000 Movie Dataset** from **Kaggle**. This dataset contains 5K data points of various movies and users. This data is stored in a matrix called the user-movie interactions matrix, where the rows are the users and the columns are the movies.

3.3.2 Functions requirement

Over this recommender system we have used a lot of different python modules to perform the various stages of processing the datasets. For that purpose, the various functions we used are :-

NUMPY

- ndim(): return the number of dimensions of an array.
- shape(): returns a tuple with each index having the number of corresponding elements.

PANDAS

- read_csv(): return the number of dimensions of an array.
- head(): It returns a tuple with each index having the number of corresponding elements.
- dropna(): This method allows the user to analyze and drop Rows/Columns with Null values in different ways.

OS

- walk(): It generates the file names in a directory tree by walking the tree either top-down or bottom-up.
- path(): It's another Python module, which also provides a big range of useful methods to manipulate files and directories.

NLTK

- PorterStemmer(): Its an algorithm used for removing the commoner morphological and inflexional endings from words

AST

- ast.literal_eval: Safely evaluate an expression node or a string containing a Python literal or container display.

SCIKIT-LEARN

- countVectorizer: CountVectorizer is a great tool provided by the scikit-learn library in Python. It is used to transform a given text into a vector on the basis of the frequency (count) of each word that occurs in the entire text.
- cosine_similarity: Cosine similarity is a metric used to measure how similar the documents are irrespective of their size.

3.3.3 Performance and security requirement

There are many recommendation systems available to provide users with personalized services. Among them, the most frequently used in electronic commerce is 'collaborative filtering', which is a technique that provides a process of filtering customer information for the preparation of profiles and making recommendations of products that are expected to be preferred by other users, based on such information profiles. Collaborative filtering systems, however, have in their nature both technical issues such as sparsity, scalability, and transparency, as well as security issues in the collection of the information that becomes the basis for preparation of the profiles. As the Content Based Filtering doesn't involves such sensitive information its completely safe to use such RS systems.

3.3.4 Look and Feel Requirements

A good user experience in search and recommendations are almost indistinguishable. Basically, search results are recommendations if we can formulate recommendations as search queries. It's an ideal solution as many websites and businesses already operate search engines in their backends and we can leverage existing infrastructure to build our recommendation system.

3.4 Summary

A typical recommender system based on matching and ranking usually has two types of modules. One is the matching module, and the other is the ranking module. The former performs a preliminary filtering of the 5,000 items to select those items given in the particular datasets.

CHAPTER 4
DESIGN METHODOLOGY AND ITS NOVELTY

4.1 Methodology and goal

Another common approach when designing recommender systems is content-based filtering. Content-based filtering methods are based on a description of the item and a profile of the user's preferences. These methods are best suited to situations where there is known data on an item (name, location, description, etc.), but not on the user. Content-based recommenders treat recommendation as a user-specific classification problem and learn a classifier for the user's likes and dislikes based on an item's features.

In this system, keywords are used to describe the items, and a user profile is built to indicate the type of item this user likes. In other words, these algorithms try to recommend items similar to those that a user liked in the past or is examining in the present. It does not rely on a user sign-in mechanism to generate this often-temporary profile. In particular, various candidate items are compared with items previously rated by the user, and the best-matching items are recommended. This approach has its roots in information retrieval and information filtering research.

To create a user profile, the system mostly focuses on two types of information:

1. A model of the user's preference.
2. A history of the user's interaction with the recommender system.

Basically, these methods use an item profile (i.e., a set of discrete attributes and features) characterizing the item within the system. To abstract the features of the items in the system, an item presentation algorithm is applied.

4.2 User Interface designs

User interface is everything that sits between what is going on in a user's mind and what numbers you can put in your user-item matrices that get processed into recommendations. So obviously the UI is critical in collecting signal. The main goal is usually to collect signal that is:

1. Predictive (Make sure you're collecting the right signal)
2. Unambiguous (The worst signals are ones that mean different things to different people)
3. Dense (The thing about collecting signal is that you want a lot of it.)

Below is a design of the interface we created using a framework.

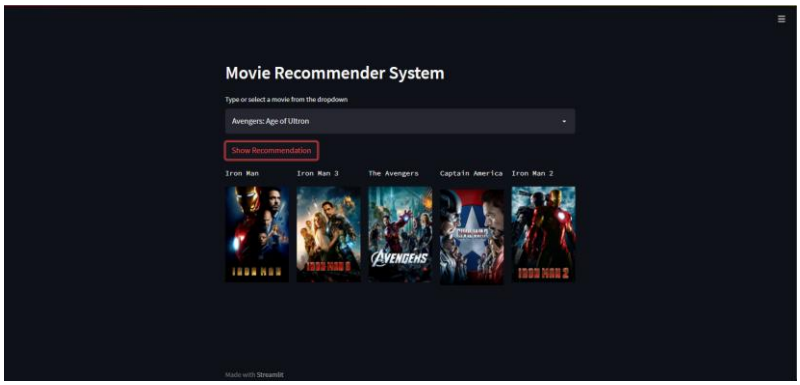

Fig 4.1-Desktop Interface

Fig 4.2-Mobile Interface

4.3 Architectural Diagram

Fig 4.3-Architectural Diagram

4.3 Summary

Recommender System (RS) is one such engine that suggests unseen items that the user may enjoy, and help them in decision making. RS helps the applications to perform better and attract users to make the applications successful which makes the RS novel.

- Example on shopping website, it keeps on suggesting the user what he/she may need to buy based on the information that the website has for the user. It actually reminds the user if he/she needed to buy something and haven't bought it yet.

- Example on Netflix/YouTube/Movie websites: It helps the user find unseen movies which he/she may enjoy.

CHAPTER 5

TECHNICAL IMPLEMENTATION & ANALYSIS

5.1 Outline

The objective of implementation analysis is the minimization of the shortfall (in output terms) or overrun (in cost terms) from the outcomes which technical analysis predicts will occur from specified input combinations. It is useful to represent the feasibility of implementation, as companion data to information on technical feasibility, in terms of a weighted index which brings together the variables germane to implementation and to present this information essentially as a probability weighting which can be applied to alternative predicted technical outcomes. In this manner selection can be based on both technical and implementational considerations.

In this phase of the report the breakdown of the project will be discussed. From breaking the problem into sub-problems and solving them in order to successfully complete the recommender system.

5.2 Step by step Implementation

5.2.1 Gather the data

We have used a public data set for movie ratings from Kaggle. The **tmdb_5000_credits.csv** and **tmdb_5000_movies.csv** contains data for about 5000 movies and has 20 columns for each movie,

```
In [37]: 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')

In [38]: movies.head(2)  #dataset for first two movies
```

Out[38]:

	budget	genres	homepage	id	keywords	original_language	original_title	overview	popularity	production_companies	product
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_
1	300000000	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	http://disney.go.com/disneypictures/pirates/	285	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	en	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	139.082615	[{"name": "Walt Disney Pictures", "id": 2}, {"...	[{"iso_

Fig 5.1 -Data Gathering Implementation

5.2.2 Data cleaning

Data cleaning is very important part of the implementation. This phase determines the efficient the execution will be and how accurate will the results be. Steps involved in the data cleaning are mentioned below:

- Checking for Missing data
- Removing duplicate data
- Resolving Null data problem
- Refining the data available in the required column
- Removing the spaces in the tags, to get the list of tags rather than a sentence
- Stemming the data or tags to remove redundancy in the tags and to improve on space management

```
In [ ]: #Keeping attributes necessary for creating tags for our data
movies = movies[['movie_id','title','genres','overview','keywords','cast','crew']]
```

```
In [ ]: #Preprocessing of data begins

#STEP 1 : Checking for Any missing data

movies.isnull().sum()        #Gives the summary of rows which have null values
```

```
In [ ]: movies.dropna(inplace=True)    #dropna() function is used to remove rows and columns with Null/NaN values.
```

```
In [ ]: #STEP 2:  Checking for any duplicated data

movies.duplicated().sum()
```

```
In [ ]: #STEP 3: Customising and Refining data to get our tags for every movie in the dataset

import ast
```

```
In [ ]: #Three Helper Functions to ease the task of customising and refining data

def convert(data):                          #The data set is in string format
    List = []                               #We need a list
    for i in ast.literal_eval(data):        #Converting the string data into a list of tags
        List.append(i['name'])
    return List
```

```
In [ ]: """
The method for converting the string data to list of tags, is same as that used for keywords and genres
But in the case for cast column, the idea is to give priority to the top 4 leading acters/actresses for recommendation
This will increase the efficiency and readability of the code(as well as the working matrix)
This is done to get the recommendation as per the first thought that the use gets when he/she hears the name of a movie
For example : If the user hears the name Iron Man, the first acter that will pop up in the user's mind will be
              'Robert Downey Jr'
"""
def top_four_people(data):                          #The data set is in string format
    List = []
    counter = 0                                     # Counter to get the top 4 cast of the movie
    for i in ast.literal_eval(data):
        if counter < 4:
            List.append(i['name'])
        counter += 1
    return List
```

```
In [ ]: """
For the case of Crew column the only need is to get the name of the director of the movie.
People usually don't remember who was the VFX expert, or who did the final editing, or who designed the sets
But people Do remember The Director in many cases
For Example, the momemnt User hears the name Justice League, the first is Snyder's Cut, Which actually gives the name
             Zack Snyder, the director of the Snyder cut

Proving Point : What was name of the head of the vfx team?
              : Like its mentioned, people dont remember :)

"""
def get_me_the_director(data):
    List = []
    for i in ast.literal_eval(data):
        if i['job'] == 'Director':
            List.append(i['name'])
    return List
```

```
In [1]: """
        STEP 4 : Transformation of data
        Before creating the tags that can be used in the recommender system, the spaces between the raw tags has to be removed
        So that no ambiguity arises when a raw tag which consists of a few words, is converted into usable tags where every word of
        the raw-tag becomes and individual tag.

        For Example : The system wants 'Science Fiction' as a one word tag 'ScienceFiction' and not as 'Science' and 'Fiction'
                      seperately.
                    : This space is problem as now 'science' and 'fiction' tags will be used even when there might not be a need.
                    : Like in case of a real autobiography of a scientist, science will be a tag, but the user who watched a
                      science-fiction movie might not be interested in real stories. And still will receive the recommendation of
                      every movie which may or may not be fiction but related to science, just because 'science' and 'fiction' tags
                      were not dealt with properly.
        Similair problem can occur when two actors sharing the first name are treated as an individual entity

        Below is the code implementation to do the task

        The lambda function used, removes the space between each element(if any) of the list supplied
        """


        movies['cast'] = movies['cast'].apply(lambda x:[i.replace(' ','') for i in x])
        movies['crew'] = movies['crew'].apply(lambda x:[i.replace(' ','') for i in x])
        movies['genres'] = movies['genres'].apply(lambda x:[i.replace(' ','') for i in x])
        movies['keywords'] = movies['keywords'].apply(lambda x:[i.replace(' ','') for i in x])

In [ ]: movies.head()         #Displaying data with the changes done to its spacing

In [ ]: """
        STEP 5 : Providing tags to our data

        Now that the missing-data problem, duplicate data problem and space problems have been resolved, its time to get the refined
        tags, which will be used by our data.
        For this a new column containing the tags will be created and the five columns that were refined previously will be merged
        into it.
        Following this step the five columns will be dropped(Not necessary but must be done), so as to improve the space utilisation
        """
        movies['tags'] = movies['overview'] + movies['genres'] + movies['keywords'] + movies['cast'] + movies['crew']

        #A new database with only three columns: movie_id,title and tags

        Movie = movies.drop(columns=['overview','genres','keywords','cast','crew'])

In [ ]: #Function to stem the tags and return the list of tags

        def stem(data):
            List = []
            for i in data.split():
                List.append(ps.stem(i))
            return " ".join(List)

In [ ]: Movie['tags'] = Movie['tags'].apply(stem)    #Stemming the tags
```

Fig. 5.2 – Data cleaning implementation

5.2.3 Modelling

In order to detect similarities between movies, data needs to be vectorized, as mentioned in the code above. CountVectorizer is used rather than TfIdfVectorizer for one simple reason: code needs a simple frequency counter for each word in tags column. Tf-Idf tends to give less importance to the words that are more present in the entire corpus (our whole column, in this case) which is not needed for this application, because every word is important to detect similarity! Once matrix containing the count for each word is achieved, the cosine_similarity function can be applied to it.

```
In [ ]: """
Now that tags are ready Vectorisation of data can be done.

The main task at hand is to find the similarities between the movies so that the recommender can act accordingly and
can provide the appropriate.

To do this the entire dataset has to be vectorised where each movie reprents a point on a 2d graph

And the recommender system will suggest the closest 'n' points, from a given point.
To convert the text to vectors, our system will be using "Bag of Words Technique".

There are other advanced techniques but because it is our first project so we chose to use a simpler yet efficient technique.

While vectorisation, the stop words are to omitted, eg: words like is,are,to etc.

For this task Scikit-learn library will be used
"""

from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer(max_features=5000,stop_words='english')

vector = cv.fit_transform(Movie['tags']).toarray()

vector.shape
```

```
In [ ]: """
For getting the cosine similarity
The definition of similarity between two vectors u and v is, in fact, the ratio between their dot product and the product of t
By applying the definition of similarity, this will be in fact equal to 1 if the two vectors are identical,
and it will be 0 if the two are orthogonal.
In other words, the similarity is a number bounded between 0 and 1 that tells us how much the two vectors are similar.
"""
from sklearn.metrics.pairwise import cosine_similarity
similarity = cosine_similarity(vector)
```

```
In [ ]: similarity[0]      #Shows The Similarity between the tags
```

Fig. 5.3 – Modelling implementation

5.3 Testing the recommender (Validation)

After all is said and done the final step in the process is to test the recommender system. The above result is very accurate, i.e. if I liked Iron Man, I will definitely love its sequel (Given that the directors kept the story consistent).

```
In [81]: recommend('Iron Man')

         Iron Man 2
         Iron Man 3
         Avengers: Age of Ultron
         Captain America: Civil War
         The Avengers
         Ant-Man
```

Fig. 5.4 – Testing the recommender system

The link to repository containing codes:
https://github.com/aryankashyap7/Movie-Recommender-System-Using-Content-Based-Filtering
Link for the website prototype:
https://mmovie-recommender-system.herokuapp.com/

5.4 Evaluation Metrics

Precision and Recall are two measurements for statistics, which are used to evaluate the quality of statistic result. Precision is used to calculate the ratio of related documents with selected documents. Recall is used to calculate the ratio of related documents with all related documents in selected documents. Below is the definition of the precision and recall under the context of our movie case. Assume *TPi* represents the number of test documents belonging to *Ci* and they are classified to *Ci* as well. *FPi* represents the number of test documents that do not belong to *Ci* are classified to *Ci*. *FNi* represents the number of test documents belonging to *Ci* are classified to other categories. So the Precision and Recall in category *Ci* is defined by:

$$P = \frac{TP_i}{TP_i + FP_i}$$

$$R = \frac{TP_i}{TP_i + FN_i}$$

Generally, we should comprehensively consider precision and recall, then we introduce F-Measure, which is defined below:

$$F = \frac{2 \times P \times R}{P + R}$$

5.5 Performance Analysis

In order to show the improvement of our TF-IIDF-DC, we split the data into 80% as training data and 20% as testing data. Both TF-IDF and TF-IIDF-DC are calculated and evaluated by Equations above. The results are shown in table 5.1

	TF-IDF			TF-IIDF-DC		
Category	P	R	F	P	R	F
Sci-Fi	86.64	83.56	85.07	89.32	85.51	87.37
Crime	78.57	75.98	77.25	85.54	81.19	83.31
Romance	70.65	67.87	69.23	76.76	69.34	72.86
Animation	78.78	77.32	78.04	85.34	81.49	83.37
Music	70.45	67.29	68.83	77.45	72.83	75.07
Comedy	80.67	75.19	77.83	88.76	82.73	85.64
War	77.87	72.87	75.29	85.91	77.34	81.40
Horror	82.34	80.17	81.24	89.21	84.65	86.87
Adventure	86.32	83.76	85.02	88.75	84.35	86.49
News	70.44	67.93	69.16	73.79	69.72	71.70
Biography	69.98	62.48	66.02	74.39	68.93	71.56
Thriller	79.89	78.28	79.08	84.18	80.38	82.24
Western	75.45	70.32	72.79	77.65	73.76	75.66
Mystery	71.22	68.95	70.07	76.85	71.45	74.05
Short	70.39	67.87	69.11	74.28	69.89	72.02
Drama	89.87	85.69	87.73	92.48	88.41	90.40
Action	80.43	76.21	78.26	84.56	81.44	82.97
Documentary	73.93	70.48	72.16	78.58	75.65	77.09
Musical	70.22	69.28	69.75	73.23	71.28	72.24
History	76.89	73.47	75.14	81.74	79.34	80.52
Family	73.49	70.34	71.88	76.43	72.67	74.50
Fantasy	73.68	71.94	72.80	77.98	74.87	76.39
Sport	77.23	73.45	75.29	80.64	75.34	77.90

Table 5.1. Comparison of TF-IDF and TF-IIDF-DC(%)

As we can see in Table 5.1 and Figure 5.5, the Precision, Recall and F-Measure by TF-IIDF-DC are all higher than that of TF-IDF. From the experiment, TFIIDF- DC strengthens the weight of representative terms and weaken terms of no use in the category, which is so-called noise.

The most important factor for content-based recommender systems is feature. How to describe a movie is the most important task because the more accurate a movie is described, the better results recommender system generates. So, from this perspective, we have proved the improvement of our approach in content-based recommender system.

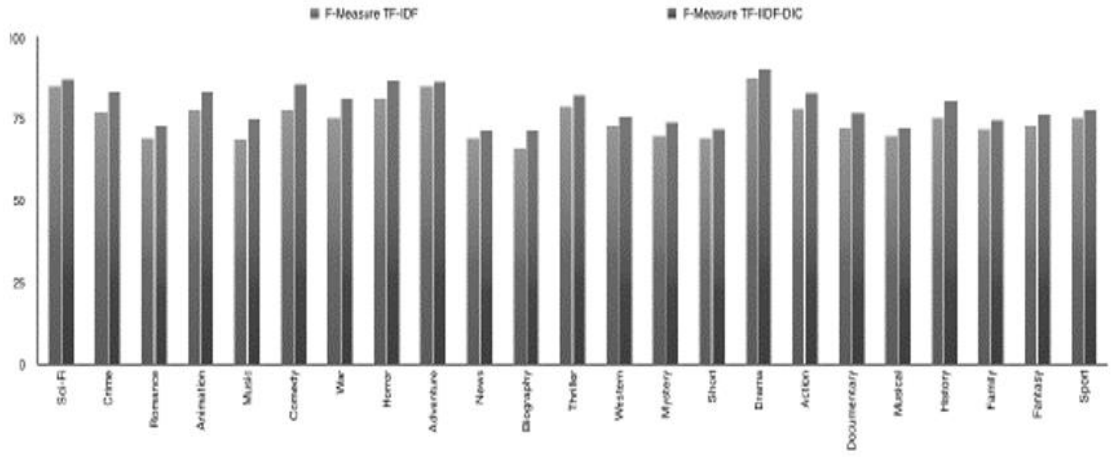

Fig 5.5 – Comparision of F-Measure

CHAPTER 6

PROJECT OUTCOME AND APPLICABILITY

6.1 Outline

Project outcomes are the changes that occur as a result of your actions. These typically involve improvements for a product or service. When designing a project, it's important to know what your project outcomes are so you have a way of measuring your success and understand what your overall goal is. In this article, we define what project outcomes are, provide several examples of project outcomes and offer steps to take when measuring yours.

6.2 Key implementations Outlines of the System

- The application is based on the content-based filtering.
- The datasets are scanned for flaws before being processed.
- Issues like missing data, duplicate data, redundant data are dealt with, successfully.
- Data is stemmed to enhance the performance and accuracy of the model.
- The application code is written using python language and is completely documented so that anyone can easily understand the working.
- The application is deployed on Heroku and can be used as an online recommender tool.

6.3 Significant Project Outcome

- The project helped the team to step into the world of recommender system.
- Gave the hands-on experience with python libraries like pandas, numpy, scikit, nltk etc.
- Helped in exploring the performance metrics used for evaluations
- Project also helped to understand good machine learning practices, like building the datasets, dividing the datasets for training and evaluations, writing modular codes etc.

6.4 Project Applicability on Real – World Applications

The application can be used as an online recommender system tool. Since the project is open-sourced, new enhancements can be made to the application based on the user's preferences. The application can be powered with the collaborative filtering techniques which in-turn is a hybrid recommending approach. With the addition of better datasets, the application can cater to recommendations in many languages as well as genres.

The application has already been deployed on Heroku, so it can be accessed from anywhere using the links mentioned before. The application provides Posters of the movies that are recommended.

6.5 Inference

Content based recommendation system is certainly good if we want to recommend suggestions based on features like genres, actors, overview etc. The accuracy of content-based filtering enhances when we use fields such as genres, cast, crews which provide more in depth information about the user's likes. Some limitations of content-based filtering are that it can only make recommendations based on existing interests of the user, it does not consider the fact that what do other users think of an item, thus low-quality item recommendations may occur sometimes.

CHAPTER 7

CONCLUSIONS AND RECOMMENDATION

7.1 Outline

Movie recommendation system in the project uses content-based filtering, this filtering first builds profiles of users and movies based on the preferences the users give or the features possessed in movies. Then, it finds matching profiles of users and movies, and recommends the unseen movies that the users may enjoy.

For example, if a user likes movies such as ‘The Prestige’ then the recommender system recommends him the movies with the genre ‘Thriller’. So, here the recommendation system tries to find similar movies to that using the information available in the database such as the lead actors, the director, genre of the film, production house, etc and based on this information find movies similar to “The Prestige”.

7.2 Limitation/Constraints of the System

The downside of a content-based movie recommender system is that the user does not get exposure to different types of movies as the recommender system recommends the movies of similar genre. Also, if the information or the data available is very less, then the prediction might not be very accurate. Therefore, the recommender system needs a large data set to make more accurate recommendation.

7.3 Future Enhancements

The project has a very vast scope in future and future enhancements. Project can be updated in near future as and when requirement for the same arises, as it is very flexible in terms of expansion. The following are some future enhancements for the project.

- Hybrid filtering can be used to provide more accurate recommendations.
- By changing the dataset and doing some modifications anyone can use the recommender system to recommend other items like songs, books, etc

7.4 Inference

Movie Recommender System is a system where anyone can write the name of the movie and related to the name, user will get recommended similar movies using content-based filtering. It is created in an easy implementation environment and provides the developer the flexibility to modify the system according to the requirements in the future.

Appendix A - vectorization

The bag-of-words (BOW) model is a representation that turns arbitrary text into fixed-length vectors by counting how many times each word appears. This process is often referred to as vectorization.

Let's understand this with an example. Suppose we wanted to vectorize the following:

- the cat sat
- the cat sat in the hat
- the cat with the hat

We'll refer to each of these as a text document.

Step 1: Determine the Vocabulary [tokenization]

We first define our vocabulary, which is the set of all words found in our document set. The only words that are found in the 3 documents above are: the, cat, sat, in, the, hat, and with.

Step 2: Count

To vectorize our documents, all we have to do is count how many times each word appears:

Document	the	cat	sat	in	hat	with
the cat sat	1	1	1	0	0	0
the cat sat in the hat	2	1	1	1	1	0
the cat with the hat	2	1	0	0	1	1

Table A – vectorization example

RELATED WORK INVESTIGATION

Recommendation Systems are a type of information filtering systems as they improve the quality of search results and provides items that are more relevant to the search item or are related to the search history of the user. **[1]** In this project we used content-based filtering as it uses item features to recommend other items similar to what the user likes, based on their previous actions or explicit feedback.**[2]** Seven different modules are used to develop the recommender system. Numpy is a general-purpose array-processing package that provides a high-performance multidimensional array object, and tools for working with these arrays. It is one of the most fundamental package for scientific computing with Python.**[3]** Pandas is a Python package that offers various data structures and operations for manipulating numerical data and time series, mainly popular for importing and analyzing data much easier.**[4]** OS module in Python provides functions for interacting with the operating system.**[5]** NLTK stands for Natural Language Toolkit and it is suite of libraries and programs in Python for Natural Language Processing Tasks. It can perform various NLP tasks like tokenization, stemming, POS tagging, lemmatization and classification to name a few.**[6]** The ast module helps to find out programmatically what the current grammar looks like.**[7]** Here it is used to convert string of list to lists. Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modelling including classification, regression, clustering and dimensionality reduction via a consistent interface in Python.**[8]** Here it is used to convert collection of text documents to a matrix of token counts and compute cosine similarity. The pickle module implements binary protocols for serializing and de-serializing a Python object structure.**[9]**

Finally, the project works as a recommender system where anyone can write the name of the movie and related to the name user will be recommended relevant movies using content based filtering from the dataset.**[10]**

REFERENCES

[1] Melville P., Sindhwani V. (2011) Recommender Systems. In: Sammut C., Webb G.I. (eds) Encyclopedia of Machine Learning. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-30164-8_705

[2] What Content-Based Filtering is and Why You Should Use It, Upwork Staff, (April 6 2021). [Online] Available: https://www.upwork.com/resources/what-is-content-based-filtering

[3] NumPy v1.21 Manual, last updated (Jun 22, 2021). [Online]
Available: https://numpy.org/doc/stable/

[4] pandas: powerful Python data analysis toolkit Release 1.3.3, Wes McKinney and the Pandas Development Team,(Sep 12, 2021).
[Online] Available: https://pandas.pydata.org/pandas-docs/version/1.3.3/pandas.pdf

[5] os module in python with examples,GeeksforGeeks, Last Updated : (09 Dec, 2021). [Online] Available: https://www.geeksforgeeks.org/os-module-python-examples/

[6] Natural Language Toolkit, 2021 NLTK Project, 3.6.5, (Oct19, 2021)
[Online] Available: https://www.nltk.org/

[7] ast — Abstract Syntax Trees, Python Software Foundation, version 3.10.1, Last updated on (Dec, 2021). [Online] Available: https://docs.python.org/3/library/ast.html

[8] Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011.

[9] pickle — Python object serialization, Python Software Foundation, version 3.10.1, Last updated on (Dec, 2021). [Online] Available: https://docs.python.org/3/library/pickle.html

[10] The Movie Database (TMDb), "TMDB 5000 Movie Dataset" version 2, Last updated (2017-09-28), Distributed by Kaggle. https://www.kaggle.com/tmdb/tmdb-movie-metadata?select=tmdb_5000_movies.csv