PICT, PUNE

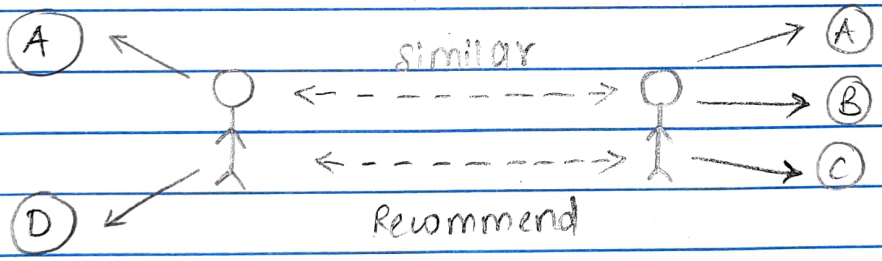
DSBDAL

MiniProject

* Title:
Movie recommendation system

* Software & Hardware requirements:
Windows OS, intel i5 processor, Jupyter Notebook

* Introduction:
A recommendation system is a simple algorithm whose aim is to provide most relevant information to a user by discovering patterns in dataset.
eg:- NetFlix / Youtube recommendations, Spotify Song recommendations.



→ Types of recommendation systems:-
(i) Content based
(ii) Collaborative based

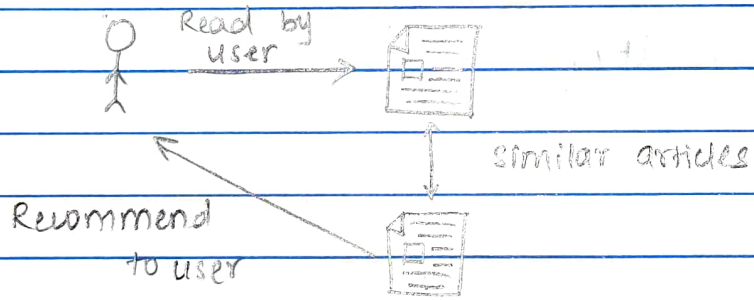
(i) Content based recommendation system:-
It uses metadata such as genre, producer, actor, musician to recommend movies. Content based systems are based on the idea that if you liked a certain item, you are most likely to like something that is similar to it.

PICT, PUNE

eg:- music recommendation for a certain artist because you liked their music.

(ii) Collaborative Filtering :-
In this type of recommendation system, the behaviour of a group of users is used to make recommendation to other users. The recommendation is based on the fact of their preferences.

→ In this project, we have implemented a content based recommendation system using scikit-learn library.



* Building the recommendation engine

1) Dataset used - movie_dataset.csv

2) After downloading the dataset, we need to import all the required libraries & then read the csv file using read_csv() method -

```
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import
                                    CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity
df = pd.read_csv("movie_dataset.csv")
```

PICT, PUNE

3) If you visualize the dataset, you will see that it has many extra info about a movie. We don't need all of them. So we choose keywords, cast, genres and director columns to use as our feature set.

```
features = ['keywords', 'cast', 'genres', 'director']
```

4) Next, create a function for combining the values of these columns into a single string

```
def combine_features(row):
    return row['keywords'] + " " + row['cast'] + " " +
           row['genres'] + " " + row['director']
```

5) Cleaning & processing the data :- fill all the NaN values with blank string in the dataframe. Then, call the above function over each row of our dataframe

```
for feature in features:
    df[feature] = df[feature].fillna('')
df['combined_features'] = df.apply(combine_features,
                                   axis=1)
```

6) Feed the combined strings to a CountVectorizer() object for getting the count matrix

```
cv = CountVectorizer()
count_matrix = cv.fit_transform(df['combined_
                                   features'])
```

7) Obtain the cosine similarity matrix from the count matrix

```
cosine_sim = cosine_similarity(count_matrix)
```

PICT, PUNE

8) Now, define 2 helper functions to get movie title from movie index & vice versa.

```
def get_title_from_index(index):
    return df[df.index == index]["title"].values[0]
def get_index_from_title(title):
    return df[df.title == title]["index"].values[0]
```

9) Our next step is to get the title of the movie that the user currently likes. Then we will find the index of that movie. After that, we will access the row corresponding to this movie in the similarity matrix. Thus, we will get the similarity scores of all other movies from the current movie. Then we will enumerate through all the similarity scores of that movie to make a tuple of movie index & similarity score. This will convert a row of similarity scores like this - [1 0.5 0.2 0.9] to this - [(0,1) (1,0.5) (2,0.2) (3,0.9)]. Here, each item is in the form - (movie index, similarity score).
For this we have created a get_recommendation function. Next we will sort the list similar_movies according to similarity scores in descending order. Since the most similar movie to a give movie will be itself, we will discard the first element after sorting the movies.
Finally, run a loop to print first 5 entries from sorted_similar_movies list.

PICT, PUNE

```
def get_recommendation (movie_user_likes):
    movie_index = get_index_from_title (movie_user_likes)
    similar_movies = list (enumerate (cosine_sim [movie_index]))
    sorted_similar_movies = sorted (similar_movies,
                key = lambda x: x[1], reverse = True) [1:]
    i = 0
    print ("Top 5 similar movies to " + movie_user_likes +
                                    " are: \n")
    for element in sorted_similar_movies:
        print (get_title_from_index (element [0]))
        i = i+1
        if i > 4:
            break
```

Output :- get_recommendation ("The Avengers")

→ Top 5 similar movies to The Avengers are:
Avengers : Age of Ultron
Iron Man 2
Captain America : The Winter Soldier
Captain America : Civil War
Thor : The Dark World

* Conclusion:
Hence we have successfully built a movie recommendation engine using scikit - learn library.