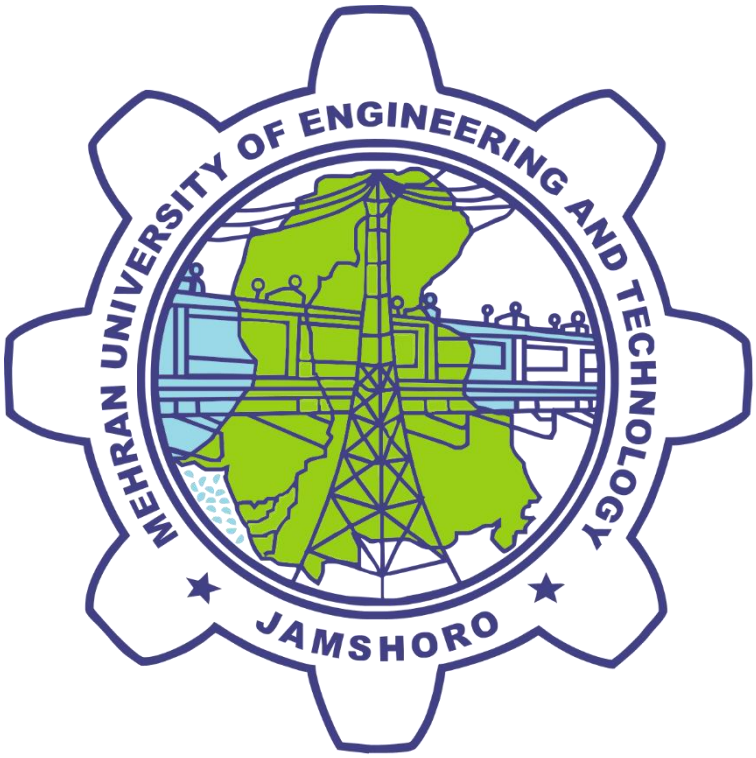
Mehran University of Engineering & Technology Jamshoro



Team: 17SW04 & 17SW52

Subject: Agent-based Intelligent Systems.

Submitted to: Ma'am Asma

Movie Recommendation Engine

Project Report

TABLE OF CONTENTS

Project Abstract

Recommendation systems are somehow very challenging tasks in Artificial Intelligence under certain constraints. In this project, we attempt to understand the different kinds of recommendation systems and compare their performance on the dataset. We attempt to build a scalable model to perform this analysis. In this project, we have used python and Machine Learning libraries like pandas and sklearn. There might be many existing systems like this but as a beginner, in Machine Learning I have used some of my ML concepts and logic to create a simple system in my own way. This is not a complete ML model but a simple python script, it is not something that you can give to a common people and they can use it but just like a python script that uses a CSV file where more than 5000 movie names are stored and then after typing your favorite movie it will recommend you some similar type of movies from that CSV file.

Discussion

Overview:

As recommendation systems are used to give customers a better experience of what is most relevant to them, the following is the mechanism of how it works:

Evaluation:

For quantifying the similarities between two or more movies we used cosine similarity which is the $\cos\theta$ between two vectors (or movies).

$$\cos\theta = \frac{\vec{u}\,.\,\vec{v}}{(|\vec{u}|.|\vec{v}|)} \qquad ,|\vec{u}|\ and\ |\vec{v}|\ are\ the\ length\ of\ vector\ \vec{u},\vec{v}$$

This cosine function is used in python script by importing CountVectorizer from the sklearn's feature_extraction library to find the similarity between movies.

We use the mean squared error metric to evaluate the predictions made by our system. The mean squared error is computed as

$$\text{MNS} = \frac{1}{N}\sum_{i=0}^{N}(pij - rij)^2$$

Where N is the number of ratings in the test partition, pij is the predicted rating for user i and movie j and rij is the actual rating.

Working:

- Pandas library is used to read the data from the CSV file
- Taking some features from the dataset like keywords, cast, director names then creating a column in a data frame which will combine all the selected features
- Passing the data frame column in CountVectorizer to find the cosine similarity
- Get the movie name from the user as an input
- Get the index of the movie that came from the user
- Pass that index in cosine similarity function to find the similar type of movies in the CSV file
- Enumerate the similar type of movies coming from a cosine similarity function into a List.
- Sort the enumerated list
- Print the result

ALGORITHMS

Algorithms for our project, we focused on two main algorithms for recommendations: Collaborative filtering & Content-based filtering.

Results / Conclusion

In last, for getting results from the algorithm we used ‘for loop’ for printing sorted values of the list.

There might be an ‘if condition’ for getting limited numbers of similar movies from a list. i.e. In this model, we want to show the 10 most similar movies to the user, so we used the condition (>= 10), etc.

Following is the output/result of the model:

```
In [1]: runfile('D:/ML Projects/Movies Engine/movie_recommender.py', wdir='D:/ML Projects/
Movies Engine')

Enter name of the Movie you like most, to see top similar movies like that one | 'Please
use a suitable name i.e. [Avatar, Gravity, Wanted, Spider-Man]' : Gravity
Top 10 similar movies to Gravity are:

The Astronaut's Wife
Space Dogs
Silent Running
Alien
Cargo
Moonraker
Planet of the Apes
Guardians of the Galaxy
Jason X
Avatar
Elysium

In [2]: runfile('D:/ML Projects/Movies Engine/movie_recommender.py', wdir='D:/ML Projects/
Movies Engine')

Enter name of the Movie you like most, to see top similar movies like that one | 'Please
use a suitable name i.e. [Avatar, Gravity, Wanted, Spider-Man]' : 3idiots
Please enter a valid movie name.

In [3]:
```

Future Works

There are plenty of ways to expand on the work done in this project. The most obvious ideas are to add features to suggest movies with common actors, writers. In addition, movies released within the same time period could also receive a boost in likelihood for recommendation. Also, there can be many other factors that can be used to provide better recommendations.

In addition, we could try to develop hybrid methods that try to combine the advantages of both content-based methods and collaborative filtering into one recommendation system.

References:

Dataset downloaded from:
https://data.world/data-society/imdb-5000-movie-dataset

Introduction to Recommender System:
http://blog.ethanrosenthal.com/2015/11/02/intro-to-collaborative -filtering/

Cosine similarity function (CountVectorizer) documentation:
https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html