PUNE INSTITUTE OF COMPUTER TECHNOLOGY,
DHANKWADI, PUNE - 411043

CLASS: TE-2

BATCH: L2

SUBJECT: DSBDAL

Group Members:

Yash Rajput (31389)

Gausiya Sayyad (31367)

Mini Project

Title: Movie Recommendation model as a mini project.

Problem Statement: Develop a movie recommendation model using the scikit-learn library in python.

Objectives:

- To understand the model building for movie recommendation.
- To explore various text analytics libraries.

Outcomes:

- To be able to perform features extraction from the text
- To be able to use nltk and CountVectorizer libraries for building recommendation models.

Requirements:

- Computer System with:
- I5 processor, 256 GB SSD, 8GB RAM.
- Jupyter Notebook
- Python with sklearn, rake_nltk, pandas, matplotlib, etc.

Introduction to Project:

We use computers to make predictions to help us achieve better results using various computational statistics. Tasks can be performed without being explicitly programmed to do so. It becomes a tedious task to extract the relevant information. Search engines solve the problem to some extent but it does not solve the personalization problem.

Recommendation System framework plays a vital role in today's internet surfing, be it buying a product from an e-commerce site or watching a movie on some video-on-demand service. In our everyday life, we depend on recommendations given by other people either by word of mouth or reviews of general surveys. People often use recommender systems over the web to make decisions for the items related to their choice.

Recommendation systems are software tools and techniques whose goal is to make useful and sensible recommendations to a collection of users for items or products that might interest them.

Concepts about Recommendation System

The recommendation systems belongs to a class of information filtering system that aims at predicting the 'preference' or 'rating' given to an item. Recommendation systems are primarily using three approaches.

1. Content-based filtering
2. Collaborative filtering
3. Hybrid filtering.

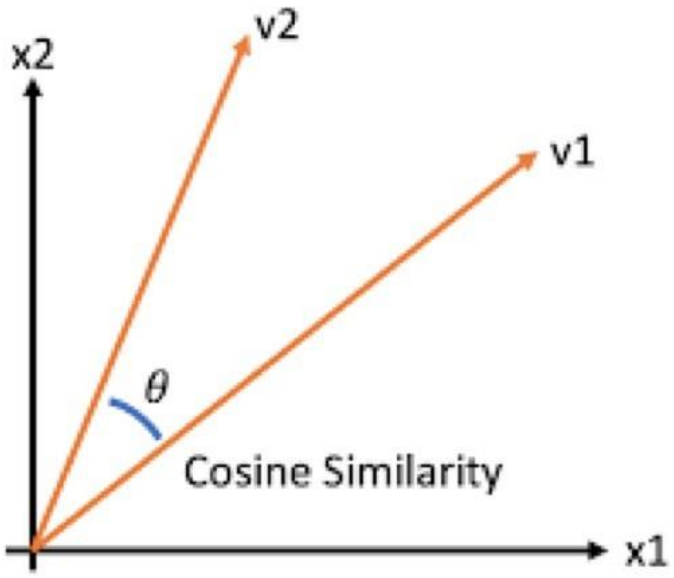
Content-based filtering: This approach filters the items based on the likings of the user. It gives result based on what the user has rated earlier. The method to model this approach is the Vector Space Model (VSM). It derives the similarity of the item from its description and introduces the concept of TF-IDF (Term Frequency-Inverse Document Frequency).

The similarity between item vectors can be computed by three methods:

1.Cosine_similarity
2.Euclidian_distance
3. Pearson's correlation

COSINE SIMILARITY

Cosine similarity among two objects measures the angle of cosine between the two objects. It compares two documents on a normalized scale. It can be done by finding the dot product between the two identities.



As the above diagram shows, the angle between v1 and v2 is. Lesser the angle between the two vectors more is the similarity. It means if the angle between two vectors is small, they are almost alike each other and if the angle between the two vectors is large then the vectors are very different from each other.

$$CosSim(x,y) = \frac{\sum_i x_i y_i}{\sqrt{\sum_i x_i^2}\sqrt{\sum_i y_i^2}}$$

This is the Cosine similarity formula which is used for the recommendation of movies.

B. Collaborative Filtering

It depends upon the users who have similar interests and gives the result based on all the users. User-based: In user-based collaborative filtering, it is considered that a user will like the items that are liked by users with whom have comparable taste.

Item-based: Item-based collaborative-filtering is different, it expects the users to like items that are related to items that he has liked earlier.

C. Hybrid Filtering

Hybrid filtering can be known as a combination of collaborative filtering and content-based filtering. It is the most common and popular technique today. It avoids the weakness of every single recommender technique.

There are some problems related to the recommendation system. They are: Cold-start problem: When a user registers for the first time, he has not watched any movie. So, the recommendation system does not have any movie based on which it can give results [8]. This is called the cold-start problem [9]. Recommendation system goes through this problem as a result of no previous record. This happens with every new user once.

Data sparsity problem: This problem occurs when the user has rated very few items based on which it is difficult for the recommendation system to give accurate results. In this problem, the results given are not much similar to the expected result. Sometimes, the system fails to give successful results and generates weak recommendations.

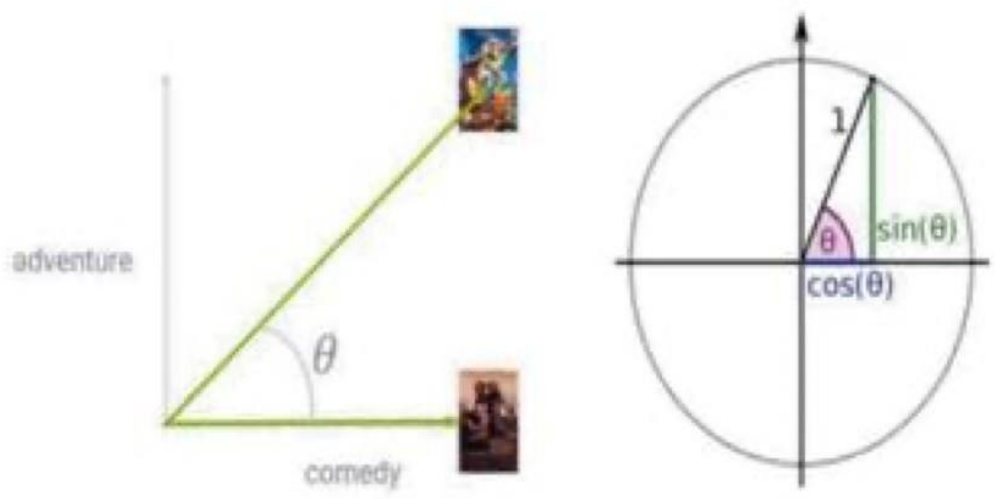


Fig: Cosine similarity

The angle theta between the two movies will determine the similarity between the two movies. The theta ranges from 0- 1. If the value of the theta is near 1 then it is most similar and if it's near to 0 then it is least similar. The movie will be recommended if it is close to 1 otherwise there would be no similarity between them. It will recommend the best movies to the user according to the Cosine similarity. After the cosine similarity, we have used a normalised popular score through which we get our function of computing distance. Then by using the KNN functionality**,** we have found the nearest neighbour which will be recommended to the user.

Implementation

Dataset Used:

- **"MoviesOnStreamingPlatforms_updated.csv",**
- **" netflix_titles.csv".**

About Dataset:

About this Dataset, netflix *is one of the most popular media and video streaming platforms. They have over 8000 movies or tv shows available on their platform, as of mid-2021, they have over 200M Subscribers globally. This tabular dataset consists of listings of all the movies and tv shows available on Netflix, along with details such as - cast, directors, ratings, release year, duration, etc.*

Dataset Attibutes:

- show_id (id of the show)
- type (type of the show)
- title (title of show)
- director (director of the show)
- cast (casting)
- country
- date_added (date added on the ott)
- release_year
- rating (imdb rating)
- duration (duration in minutes)
- listed_in (genre of the show)
- description (description of the show)

Libraries used:

- pandas
  - matplotlib
  - pandas_profiling
  - warnings
  - collections.
  - sklearn.metrics.pairwise.cosine_similarity
  - from sklearn.feature_extraction.text.CountVectorizer
  - nltk_rake

pandas_profiling: The pandas_profiling library in Python include a method named as ProfileReport() which generate a basic report on the input DataFrame.

The report consist of the following:

- DataFrame overview,
- Each attribute on which DataFrame is defined,
- Correlations between attributes (Pearson Correlation and Spearman Correlation), and
- A sample of DataFrame.

Nltk_Rake

Rake also known as **Rapid Automatic Keyword Extraction** is a keyword extraction algorithm that is extremely efficient which operates on individual documents to enable an application to the **dynamic collection**, it can also be applied on the new domains very easily and also very effective in handling **multiple types of documents**, especially the type of text which follows **specific grammar conventions**.

Rake is based on the observations that keywords frequently contain multiple words with **standard punctuation** or **stop words** or we can say functioning words like **'and', 'of', 'the'**, etc with **minimum lexical meaning**. Stop words are typically dropped within all the **informational systems** and also not

included in various text analyses as they are considered to be meaningless. Words that are considered to carry a meaning related to the text are described as the content bearing and are called as content words.
The input parameters for the RAKE Algorithm comprise a list of stop words also a set of phrase delimiters and word delimiters. It uses stop words and phrase delimiters to partition the document into candidate keywords, these candidate keywords are mainly the words that help a developer in extracting the exact keyword necessary to get information from the document.

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. This is helpful when we have multiple such texts, and we wish to convert each word in each text into vectors (for using in further text analysis).
CountVectorizer creates a matrix in which each unique word is represented by a column of the matrix, and each text sample from the document is a row in the matrix. The value of each cell is nothing but the count of the word in that particular text sample.

Observations

- The MoviesOnStreamingPlatforms_updated.csv dataset has been used for exploratory analysis for movies across all platforms.
- Among 16744 movies, Lagaan movie has longest runtime.
- Among movies from several platforms , the amazon prime has highest number of 8+ imdb ratings.
- For Netflix_titles dataset, the movie recommendation model is built.
- The rake library extracts keywords from description of the movie and make a keyword score by appending with director, genre, cast, etc details.
- This keyword score is passed to CountVectorizer for model building.
- The recommendation model built is use this keywords to recommend related movies in list.
- Array of movies with similar keywords are displayed as recommendations.

Output

```
To exit Enter "quit"

Enter The Name of a Movie or Tv Show: 3 idiots
['PK', 'Moms at War', 'Pahuna', 'Acapulco La vida va']
Enter The Name of a Movie or Tv Show: inception
['Apollo 18', 'Forbidden Planet', 'Limitless', 'The Darkest Dawn']
Enter The Name of a Movie or Tv Show: irishman
The movie or Tv Show does not exist

Enter The Name of a Movie or Tv Show: the irishman
['The Irishman: In Conversation', 'Pulp Fiction', 'Rolling Thunder Revue: A Bob Dylan Story by Martin Scorsese', '
Taxi Driver']
Enter The Name of a Movie or Tv Show: quit


'''Thank you'''
```

Conclusion

We have illustrated the modelling of a movie recommendation system by making the use of content-based filtering in the movie recommendation system. The KNN algorithm is implemented in this model along with the principle of cosine similarity as it gives more accuracy than the other distance metrics and the complexity is comparatively low too Recommendations systems have become the most essential fount of a relevant and reliable source of information in the world of internet.