BUILDING A MOVIE RECOMMENDER SYSTEM WITH TMDB_5000 DATASET

By – Sayan Das

JUNE 1, 2023

Contents

Introduction

In today's era of information overload, finding personalized and relevant movie recommendations has become increasingly challenging. The vast amount of available movies makes it essential to have a system that can effectively filter and suggest movies based on individual preferences. To address this need, we present our movie recommendation system, a project that leverages Natural Language Processing (NLP) techniques to process the contents of the movie dataset to build a content-based recommendation system for movies.

The main objective of this project is to offer tailored movie recommendations to users by analysing the textual content associated with movies. Unlike collaborative filtering methods that rely on user behaviour and preferences, a content-based approach focuses on the characteristics of movies themselves. By utilizing NLP techniques, the project will extract meaningful information from movie descriptions, genres, tagline, casts and other textual features to create an accurate representation of each movie.

To achieve this, the project involves an extensive data pre-processing pipeline, where the textual content is transformed into a structured format that can be effectively analysed by NLP algorithms. Techniques such as tokenization, lemmatization, stop-word and punctuations removal will be employed to clean and transform the raw text data.

Once the pre-processing is complete, the recommender will create a comprehensive movie profile that encapsulates the important features and attributes of each movie. These profiles will then be used to calculate similarity scores between movies, enabling the system to recommend movies that have similar content. Users will have the ability to input their preferences, which will be matched against the movie profiles to generate personalized recommendations.

This project aims to provide several benefits to both movie enthusiasts and streaming platforms. Users will receive accurate and personalized movie recommendations that align with their preferences, leading to an enhanced movie-watching experience. Streaming platforms, on the other hand, can leverage this recommender to improve user engagement, retention, and overall customer satisfaction.

The Dataset

For the purpose of building the recommendation system we are going to use the tmdb_5000_movie_dataset. The dataset contains details of the information for about 5000 movies available. This dataset is available in Kaggle website in the following link

https://www.kaggle.com/datasets/tmdb/tmdb-movie-metadata

This dataset contains 2 csv files named as –

1. tmdb_5000_movies.csv
2. tmdb_5000_credits.csv

Both of these two dataset contains 4803 rows.

The columns of the tmdb_5000_movies.csv are described below –

1. **budget** – budget in US dollar of the movie

2. **genres** – words describing the genre of the movie
3. **homepage** – link to the website of the movies
4. **id –** the unique id of the movie in tmdb database
5. **keywords** – keywords for the movie
6. **original_language** – original language of the movie
7. **original_title** original title of the movie
8. **overview** – brief text description of the movie
9. **popularity** – popularity score of the movie calculated by tmdb
10. **production_companies** – name of the production companies associated with the movie
11. **production_countries** – country of production
12. **release_date** – date of release of the movie
13. **revenue** – revenue in US dollar from the movie
14. **runtime –** total runtime in minutes of the movie
15. **spoken_languages** – language spoken in the movies
16. **status** – whether it is released or not
17. **tagline** – tagline of the movie
18. **title –** title to the movie
19. **vote_average** – average votes of the movie
20. **vote_count** – total votes received by the movie

on the other hand the columns of the tmdb_5000_credits,csv has the following column description –

1. **movie_id** – unique id of the movies assigned by tmdb database
2. **title –** title of the movie
3. **cast –** casts of the movies
4. **crew –** crew of the movie

We can use this dataset to build up the recommendation system that we want.

Recommendation System

In the dataset information like genres of the movie, casts, directors, movie keywords etc. are given for each of the movies. We can use this data to get information about the contents of all the movies in the dataset. All this contents that are available in the dataset are suggesting to build up a content-based recommendation system. Let us take a look at what is a content-based recommendation is.

Content-Based Recommendation System

The main idea of these recommendation is described in a sentence “find out the most likely movie and recommend it”. That is, given a movie, find out, which are the others movies that has similar contents as the given movie and recommend those other movies to the user.

Here from the text-descriptions of the movies, which are in the dataset we have to figure out the main contents of the movies and represent it numerically. Every movie will have such a numeric representation. Next, we have to prepare the similarity scores based on the numeric representation, previously obtained, among every pairs of movies in the dataset. At the end of doing this we get a matrix of similarity scores. Now, given a movie we have to look for

movies which has the highest similarity scores with the given movie. Then we can recommend the 1st 10 or 20 of those movies to the users.

Note that since the recommendation is based on the similarity scores of the content of the movies, the recommended movies are similar with respect to their content. The given dataset is ideal for the content-based recommendation. We are going to use this kind of recommendation in the project

Data Pre-processing

Before we go into the analysis of the movie dataset we need to pre-process the dataset, so that is useful to process. We have gone through the following stages to pre-process the dataset.

Merging the dataset

In the first step we are going to merge the two datasets to get all the information in a single csv file. We have done it by using the merge function, applying the inner join techniques. We used the id from the movies dataset and the movies_id (these two are basically the same columns) from the credits dataset to merge the two dataset.

The new dataset now contains 20+4 = 24 columns and 4803 rows.

Discarding the unimportant columns

In the next step we have deleted the columns that are not actually required for the purpose of building a movie recommender. We have deleted the following columns from the dataset-

1. budget
2. homepage
3. original_title
4. production_countries
5. revenue
6. spoken_languages
7. status
8. title_y
9. movie_id

After deleting these unnecessary columns, we are left with the following columns-

1. genres
2. id
3. keywords
4. original_language
5. overview
6. popularity
7. production_companies
8. release_date
9. runtime
10. tagline
11. title
12. vote_average
13. vote_vount

14. cast
15. crew

We are going to build our recommender based on these 15 columns.

Discarding the missing observations

After the exploring with the columns that are left, we have found out that there are missing values in the columns

1. overview – 0.06% missing values
2. release_date – 0.04% missing values
3. runtime – 0.02% missing values
4. tagline – 17.57% missing values

we have deleted all the rows that contains missing values in the first 3 columns. We replaced the missing values in the tagline column by a blank text like ‘’. Every time we add the tagline features for any rows that have missing tagline, it will consider those blank text.

Extracting information from some specific columns

Some of the columns, like genre, keyword, production_companies, casts or crew; have stored information in a special way. One example for genres is like

[{"id": 12, "name": "Adventure"}, {"id": 14, "name": "Fantasy"}, {"id": 28, "name": "Action"}]

It is stored as a collection dictionaries in a list. We have to extract all the values corresponding to the key ‘name’ from all of the dictionaries in the list.

We have done this extraction for the columns genres, keyword and production companies.

As for the cast, the name of the casts of the movie is given according to their important. For a particular movie the main casts are given in the first place. We have extracted only the 1st 3 casts.

From the crew section we have only extracted the name of the directors of the movies.

Building up the content as text

Next, we have built up a single text for each of the movies by concatenating the genres, keywords, overview, production_companies, tagline, cast and director for the corresponding movies.

This single text contains all the keywords, contents that are relevant to the movie. We can consider it as a content to the movie. We stored it in a new columns ‘content’, of the dataset.

Extracting the features

Now the only pre-processing step that we are left with is the feature extraction. We have built up a text content that carries all the relevant words to the corresponding movie. But this is in the form of text. We cannot calculate the similarity scores between two movies using this text format. To make it enable for the calculation of the similarity score we need to convert it to the numerical vectors.

We have converted the text to numerical vectors using the TF-IDF vectorizer. Lets take a look on how this actually works.

TF-IDF Vectorizer:

TF-IDF stands for Term Frequency-Inverse Document Frequency. It is a statistical measure that is used to quantify the importance of a term or word in a document within a collection of documents. TF-IDF is calculated as follows:

$$\text{TF-IDF} = (\text{Term Frequency}) * (\text{Inverse Document Frequency})$$

Where,

Term Frequency = number of times a term appears in a document.

Inverse Document Frequency = logarithm of the number of documents in the collection divided by the number of documents that contain the term.

TF-IDF is a good choice for these tasks because it is able to capture the importance of terms in a text content, while also taking into account how common those terms are in the overall collection of text contents.

We have used this vectorizer to convert the text content into numerical vectors of length 5000 (we have set the maximum feature to 5000). Now, the content of each movie in the dataset is represented by the vectors of equal lengths.

Building Up the Recommendation System

Now that we have the contents of each of the movies in terms of the vectors of equal lengths, we can calculate the similarity score between each pairs of the movies and store this similarity score in terms of a matrix $R = ((r_{ij}))$, where r_{ij} denotes the similarity score between i^{th} and j^{th} movies. The i^{th} row of the matrix R represents the similarity scores of the i^{th} movie with all other movies in the dataset.

These similarity score in our projects is calculated using cosine similarity. The value of the lies between -1 and 1. The higher the similarity scores are between two movie, the more the similar the movie is with the other movie, with respect to its content. Any movie has the similarity score 1 with the same movie.

Now given a movie by the user, we can find out, which row in R represent that particular movie. Say the k^{th} row. Then we isolate the k^{th} row. This row represents the similarity scores of the given movie with all other movies. We arrange this row with the descending Order of magnitude. It will bring the movie with highest similarity score in the first place. We can suggest the first ten or twenty movies from this list as a recommendation.

Some examples

We have implemented the above-mentioned algorithm to build up a content-based recommendation. Let try out inputting some movies and find out the result

Example 1

INPUT - Pirates of the Caribbean: At World's End

It’s a pirate adventure movie.

Recommendation:

1. Pirates of the Caribbean: Dead Man's Chest
2. Pirates of the Caribbean: The Curse of the Black Pearl
3. Pirates of the Caribbean: On Stranger Tides
4. VeggieTales: The Pirates Who Don't Do Anything
5. Life of Pi
6. The Pirates! In an Adventure with Scientists!
7. Shipwrecked
8. 20,000 Leagues Under the Sea
9. Nim's Island

All the recommendations are either related to pirates or sea

Example 2

INPUT – Iron Man

It's a superhero movie from the "Marvel Studios" Production House

Recommendation:

1. Iron Man 2
2. Iron Man 3
3. Avengers: Age of Ultron
4. Ant-Man
5. The Avengers
6. Thor
7. Thor: The Dark World
8. Captain America: The First Avenger
9. Captain America: Civil War

All the recommended movies are superhero movies from the marvel studios

Example 3

INPUT – Harry Potter and the Chamber of Secrets

It's a magical fantasy movie.

Recommendation:

1. Harry Potter and the Prisoner of Azkaban
2. Harry Potter and the Order of the Phoenix
3. Harry Potter and the Philosopher's Stone
4. Harry Potter and the Half-Blood Prince
5. Harry Potter and the Goblet of Fire
6. The Craft
7. Practical Magic
8. Night at the Museum: Secret of the Tomb
9. Something's Gotta Give

All the recommendations are either other movies of Harry Potter series or related to magical fantasy.

Conclusion

By analyzing the characteristics and attributes of movies, these systems can provide personalized recommendations that align with our individual tastes and preferences. The main advantage of content-based approaches is their ability to suggest movies based on their inherent features, such as genre, director, actors, and plot, without relying on user behavior or historical data.

Furthermore, content-based recommendation systems offer increased transparency and interpretability. The recommendations are generated based on explicit features of movies, allowing users to understand why certain suggestions are made. This transparency fosters user trust and engagement, as they can comprehend the underlying logic of the system's recommendations.

However, content-based approaches also have limitations. They rely heavily on the accuracy and quality of metadata associated with movies, such as genre tags or actor information. Inaccurate or incomplete metadata can lead to suboptimal recommendations. Additionally, content-based systems may struggle to capture complex user preferences or recommend movies that deviate from a user's usual choices.

In summary, content-based movie recommendation systems have significantly enhanced the way we discover and enjoy films. They enable personalized suggestions based on movie characteristics, mitigate the cold-start problem, and offer transparent recommendation processes. While they have their limitations, integrating content-based approaches with other methods can lead to even more robust and effective recommendation systems in the future.

We have built up a flask application using, based using the above recommendation. The links to all the codes are given.

References

The links to the dataset and the codes are as follows –

1. Dataset Link: https://www.kaggle.com/datasets/tmdb/tmdb-movie-metadata
2. Project Link: https://github.com/sayandas1302/Movie-Recommendation-System