Movie Recommendation: A Machine Learning Project

November 9, 2021

A. Introduction

A1. Background

Stuck not knowing which movie or series to watch after binge-watching the latest show? Then this project is just for you! Honestly, with so many options to choose in Netflix there should be no problem having to know which movie to watch next. But it is! Ever wondered how Netflix chooses movies to put under the "Recommended for you" section? Or how it picks the movies to suggest when the credits hits? I am profoundly curious as well!

While this project may not be the exact algorithm Netflix or other media subscriptions use in their own sites, this is just one way (of the many available) to go about solving this curiosity. In this project, we would a data-driven approach to quench the thirst of this interest. We would use, not in particular a machine learning method, but a simple algorithm that has been widely used in many fields: Cosine Similarity.

A2. Problem

For this project, the question that can be asked: "How to build a movie recommendation system using Cosine Similarity Algorithm?"

A3. Target

The intended audience for this project are people who are movie-goers that has no subscription to sites like Netflix that could recommend them movies or series to watch next. Maybe this project could be a way for them to make a system that could easily be built using any programming software they may have. This would give them a list of movies to watch based on the content of their preferences and popularity of the options.

B. Data Acquisition and Cleaning

B1. Data Source

The following data is used:

1. Dataset that contains information about each movie in Hollywood:

Data Source = Siddhardhan:
https://drive.google.com/file/d/1cCkwiVv4mgfl20ntgY3n4yApcWqqZQe6/view

Description: The dataset contains 4803 entries and 24 attributes.

Matrix column attributes:

- Index, Budget, Genres, Homepage;
- ID, Keywords, Original Language, Original Title;
- Overview, Popularity, Production Company, Production Countries;
- Release Date, Revenue, Runtime, Spoken Languages;
- Status, Tagline, Title, Vote Average;
- Vote Count, Cast, Crew director

B2. Data Cleaning

The dataset initially has to be downloaded in the data assets column in Watson Studio. The file has 4803 entries and 24 attributes.

As part of the objective of this project, we want to incorporate content-based criteria to make a recommendation system. To do this, we initially selected the relevant features for the proposition. We ended up picking only five features: “genres”, “keywords”, “tagline”, “cast”, and “director”. These features were chosen as it best describes the content for each movie. We will use the values for these columns in order to calculate Cosine Similarity Algorithm.

After the features were chosen, any null values in each feature were then replaced with null string. Then all values of the five selected features were combined to form one column. This column of combined features was then converted from text data to feature vectors. This is done in order to be able to use these text values as input for the algorithm.

TF-IDF Vectorizer was the function used to convert the text data to feature vectors. This is a measure of originality of a word by comparing the number of times a word appears in a document with the number of documents the word appears in. It returns a document-term matrix that will be used as an input to calculate for the cosine similarity.

Cosine similarity measures the cosine angle between two vectors projected in a multi-dimensional space. It is a metric used to determine how similar the documents are irrespective of their size. This is a great algorithm to use as the Euclidean distance of two vectors in a document is large, as long as there are many common words shared both would still have a small cosine angle. The smaller the angle is, the more similar the two vectors are. This algorithm would return a matrix that has the same number of rows and columns.

C. Exploratory Data Analysis

Now that most of the process is out of the way, we can start on taking input from the user and build a system that would put together all the algorithms used. First, the input from the user must be stored in a variable. This variable would then be used throughout the system.

We then created a list with all the movie names given in the dataset. The list would be used to find a close match for the movie name given by the user. To do this, a module named "difflib" was used. It has a function that returns a list of the best "good enough" matches using two parameters: the variable used to store input of user and the list of all movie names. The first movie title returned by the function can then be used to find the index. It is important to note that only the first variable of the output is necessary as it is the bears the closest match to the user input.

After the index was found, the similarity score between the specific movie and every other movie in the dataset was calculated. This returned an array that contained the index of the movie as its first value and the similarity score as the second value. To evaluate which movie has a high similarity score with the user input, we can sort the similarity score in a decreasing order.

Last step is to print the name of similar movies based on the index. The number of movies to list can be limited to the number wanted by the user.

C1. Building a Movie Recommendation System

Building a recommendation system requires asking the user for two inputs. The first being a question that would ask the user if they would want to attain a movie recommendation. This input would drive the entire system. Meaning that as long as the user would reply a "yes", the movie recommendation system would continue to recommence after returning a list of movies to suggest.

The last query the system would ask is the number of suggestions the user would like and the name of the movie separated by space. If the title of the movie is in the dataset, then the whole process above would commence. If not, then the system would tell the user that the movie is not in the dataset and would then ask the user for a new input. When the input is in the dataset, the system would the return a list of recommendation.

Finally, as stated, after returning a list of prescribe movie the system would then ask the user if they would want a new recommendation or not. If the user answers "no" then the system would conclude.

D. Conclusion

Now that there is a system that does not require a fancy method to solve the interest at hand, one can build there own at their disposal. It does not require a fancy programming software as well! Though an extensive list of movies might be hard to personally gather, anyone can use datasets that is easily accessible online. This whole project is designed to guide anyone who

would want to make their own recommendation system. Hopefully, this report is extensive in sharing my work.