CST4802; Final Project Report

With the chosen project topic on interest; building a recommender system using content-based filtering, I have successfully implemented a recommender system that suggests the next top five following movies based on the user's preferences or past interactions based on the similarity of movie to movie (to be specified). The implementation of this project is done within Google Colab, utilizing the programming language; Python, the dataset found via GitHub provided by Reisner (the GitHub user), and the file; HollywoodMovies.CSV (CSV format). As for the following steps to implement the project task, I have loaded the dataset from GitHub using the URL for convenience instead of downloading and uploading it onto my Google Colab notebook. Utilizing the imports of pandas allowed me to read the dataset via the URL.

Although not getting familiar with the dataset could not be as ideal, therefore I would not know what has been duplicated, what columns aren't needed, and if the data is fully usable, such as no missing/null/NaN values, I will clean the dataset that I am using, in other words, to preprocess the dataset, getting it ready for our main project task and before, preprocessing, deciding what columns not necessary to make an impact/use in our task implementation and to remove those unnecessary columns to create a more presentable use. In our next upcoming new specified dataframe, the remaining columns were put to use. While preprocessing our dataset, I have checked for any null values, duplicates, etc. And to the output, yes, the dataset that I'm using is not fully preprocessed and or ready to be used on its own, therefore, I went ahead and removed the duplicated rows, and with the null values I replaced the null values, substituting it to be 'unknown' as this will not affect any of our analysis and or goal of the task; building a recommendation system for recommending the top five movies and keeping the dataset as it is; to be more thorough.

To further elaborate, I filled missing values as unknown, in dealing with missing/null values; filling missing values as "unknown" instead of leaving them as it is/using an empty string, helps us keep an eye on where data is missing, rather than treating it as if within our dataset, or considering them as 'empty data'. Reflecting null values as 'unknown" allows recognition of missing values for this particular dataset I'm using for this task. With this approach, it makes it easier to identify and handle these missing/empty values later in the process, contributing to a more consistent and manageable dataset.

On the other hand, when dealing with columns with empty/null and or NaN values that are in the numeric value category, since some fields consist of numeric values; the approach to handling missing values, to filling missing values with the mean for numerical columns including the following; RottenTomatoe and Budget helps maintain the distribution of the data. The mean is often used because it represents the average value, making it a reasonable estimate for missing data in a dataset. Filling missing values with the mean aid to prevent errors from occurring due to incomplete data while keeping the overall data we are using intact. Further to the preprocessing procedures, I have checked for duplicated rows, which were found to have duplicated rows, therefore, to be dropped/removed.

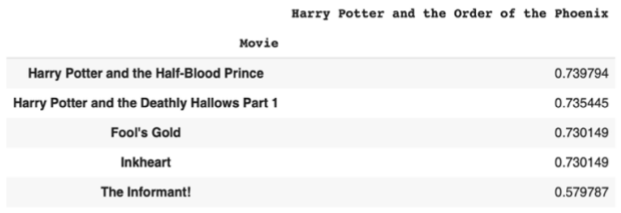
Upcoming, this part may be optional, but it's still implemented, normalizing numerical features helps ensure that different values, like RottenTomatoes, AudienceScore, and Budget columns, are on the same scale. (just so that these fields do not exceed a range, upscaling different values as an extra procedure) By scaling all features to a similar range, we make sure each category has the same equal impact on the model we are implementing (movie recommendation system), leading to more accurate recommendations for the user.

Upon the proper combination of features for our recommendation, I have combined several features to ensure the best recommendation system possible, such features combined, include Genre;(captures the type of movie genre and helps match movies of similar genre type), Story;(identify the type of movie), LeadStudio;(provides information on production studio), and RottenTomatoes;(recommend movies with similar audience or rating/critics), as I believe the implementation depends on these features to provide the most similar movies/based on what can be used among this dataset features. I skipped out using the year because I don't think that it is fair to take in year to be in effect when recommending movies, I believe when recommending the year of any movie shouldn't matter as it can be from any timeline/year of a movie.

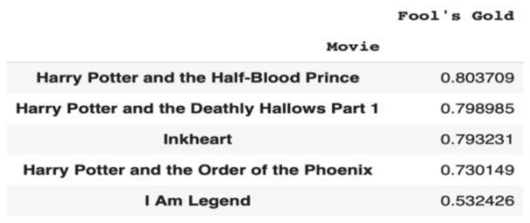
The most important part/gist of our recommendation system, implementing TF-IDF and Computing Cosine Similarity, turning movie details, like the genre, story, etc. into numbers (extracting meaningful features through text). Using TF-IDF technique (in long; Term Frequency-Inverse Document Frequency) to identify important words and phrases, thus ignoring stop words. By looking at both single words and word pairs, it creates a numerical representation for each movie based on its description, ensuring terms that appear in at least 5 movies are considered. This helps/acknowledge which movies are similar to each other, making defined recommendations. Regarding cosine similarity, it calculates how similar each movie is to every other movie using combined features like genre and story. It does this by comparing the movies based on their numerical representations, with cosine similarity measuring how closely related the two movies are. A higher similarity score means the movies are more alike. This makes it easy to see which movies are most similar, helping to recommend movies that are alike based on their descriptions.

Our next upcoming defined function recommends movies that are similar to a prompt/user-based movie based on combined features (genre & story). First, it checks if the movie title is in the dataset to be specified. Then, checks the similarity scores of that movie compared to all others, sorting them from most similar to least similar. The function excludes the original movie from the list and returns the top-recommended movies based on their similarity scores. This helps users find movies that are closely related to the one they like or are interested in.

Coming to a short end, of our result/testing whether our recommendation works or not, with the upcoming pictures provided, the output of the code provides you with a list of sorted recommended movies, and their cosine similarity score (top 5):



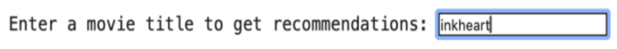
	Harry Potter and the Order of the Phoenix
Movie	
Harry Potter and the Half-Blood Prince	0.739794
Harry Potter and the Deathly Hallows Part 1	0.735445
Fool's Gold	0.730149
Inkheart	0.730149
The Informant!	0.579787



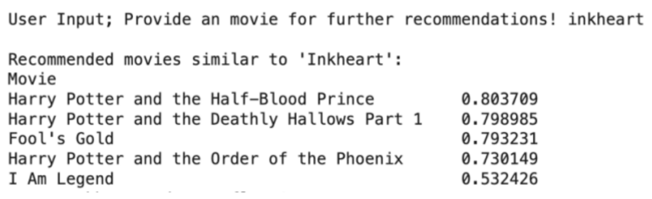
	Fool's Gold
Movie	
Harry Potter and the Half-Blood Prince	0.803709
Harry Potter and the Deathly Hallows Part 1	0.798985
Inkheart	0.793231
Harry Potter and the Order of the Phoenix	0.730149
I Am Legend	0.532426

With the output of these two examples, coded within; specifying the movie names, it shows the top 5 movies that are closely associated with the prompt movie to be given.

Upcoming, next recommender system based on user input;



```
Enter a movie title to get recommendations: inkheart
```



```
User Input; Provide an movie for further recommendations! inkheart

Recommended movies similar to 'Inkheart':
Movie
Harry Potter and the Half-Blood Prince         0.803709
Harry Potter and the Deathly Hallows Part 1    0.798985
Fool's Gold                                    0.793231
Harry Potter and the Order of the Phoenix      0.730149
I Am Legend                                    0.532426
```

Overall, both methods, whether you're specifying the movie name, or having the user input the movie name, will provide the result in a sorted manner, where movies are recommended to be the closest association with the input, or specifying it. As for a note to myself, is to revisit the work, to further analyze the dataset, come out with new data analysis, what can be found, and how to utilize the dataset, to come up with new insights.