Movie Similarity Overview

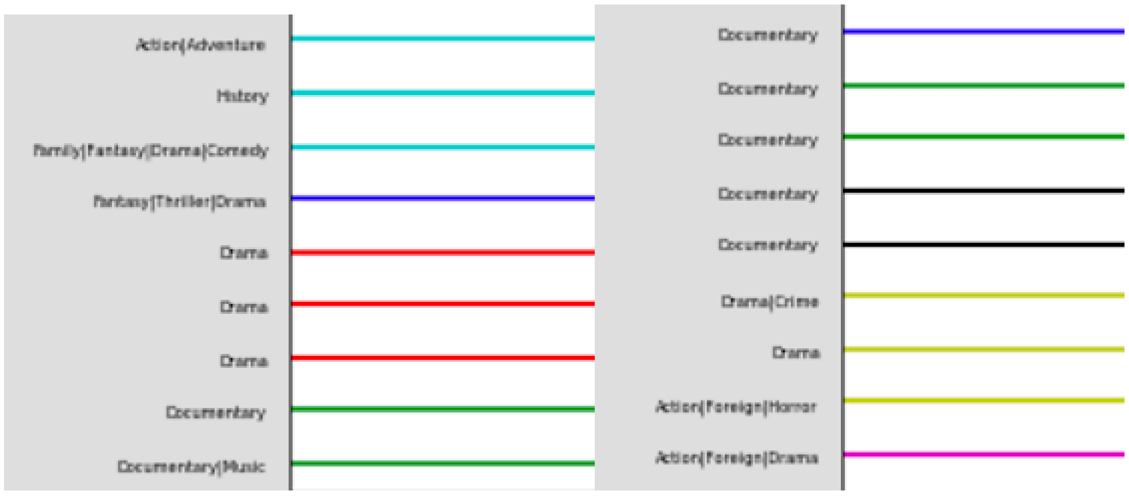
MOVIE RECOMMENDATION - DATA CLEANING

For data cleaning, missing values were removed and not included within the analysis while genres were turned into a one hot encoding format. From this cleaned data set provided, many missing values were present in the overview and genres columns, so rows with missing values in either of these columns were removed. In addition to this, a few values in the overview columns were labeled as “no overview” so these values were removed as well. Before calculations, the observation column was handled through tokenizing, stemming, and accounting for stop words. Overall, the data set was reduced from approximately 330,000 observations to 177,000 observations.

MOVIE RECOMMENDATION - EXPLORATORY DATA ANALYSIS

For exploratory data analysis regarding the creation of a movie recommendation function, we first understood that we wanted to create the function based off of document similarity from the descriptions provided in the movies data set. To do this, we initially planned on creating a document frequency matrix and using that to create a distance matrix based on cosine similarity to determine which documents were in closest proximity, description-wise, to each other; however, we figured a term frequency-inverse document frequency matrix would better account for similarity due to its priority for “rare” words (since a TF-IDF scales frequency... i.e. the more times a word shows up in all the documents, the less weight it would have). In addition to this, we ended up using a similarity matrix (just cosine similarity) rather than the distance matrix though both can be used. Do note, if one decides to use a distance matrix, then 0 would be the most similar document as opposed to 1 within a similarity matrix.

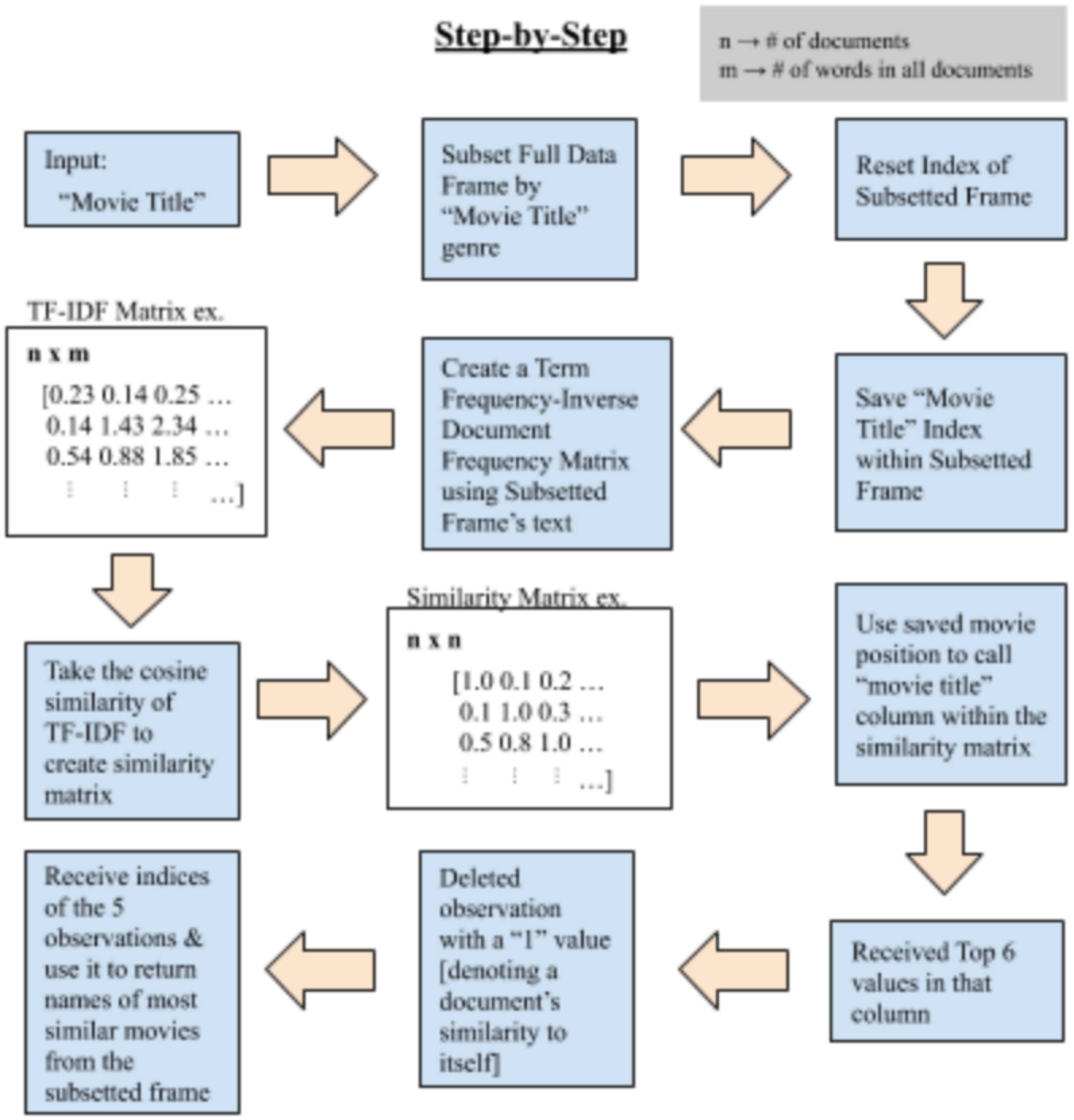
One hurdle that came up while working with a TF-IDF matrix was that this data set, with 177,000 observations would produce a TF-IDF matrix with roughly 9 million stored elements. With that many elements, producing a similarity matrix would be computationally extensive; therefore, we sampled 100 observations from the full data frame to explore a way to split the data by genre. With this sample, we created a term frequency-inverse document frequency matrix, which gave us scaled word frequencies of each document. From the TF-IDF, we calculated a distance matrix as 1 minus the cosine similarity of the TF-IDF. With this distance matrix, we used the ward clustering method and plotted the clusters as a dendrogram while labeling each observation by its genre. After creating a hierarchical clustering dendrogram, we were able to see that same-genres tended to cluster together, which then gave us a safe assumption to calculate the movie recommendation function based on similar-genres. In short, this gave us the idea to have the function recommend movies of the same genre, which would ultimately help us reduce the number of elements in the TF-IDF matrix to be able to calculate the distance matrix.



[Areas were screenshotted for visibility. See *Figure 13: Hierarchical Clustering Dendrogram* in Supplementary Report]

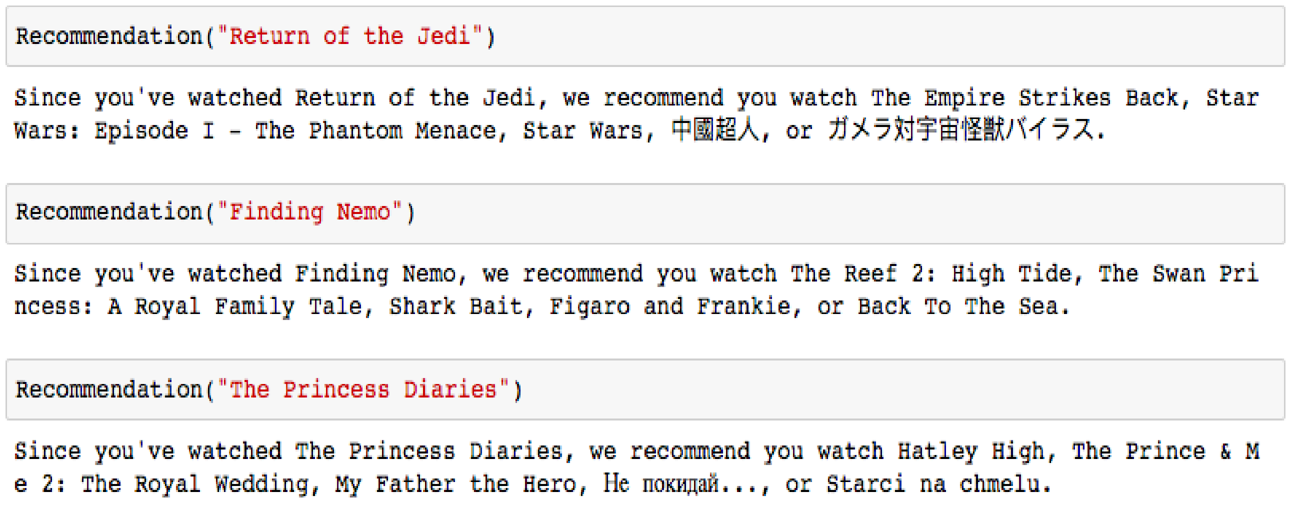
MOVIE RECOMMENDATION - PROCESS

As mentioned, the movie recommendation function will split up movies by genre; therefore, if the inputted movie was Action/Fantasy, the function would output the most similar movies, description-wise, that are also Action/Fantasy. To do this, we created a function that would take in a string (movie title), find the desired movie's genre, and subset the full data frame for that specific genre. After we subset, we initially used count_vectorizer() and .fit_transform to create a document term matrix; however, a term frequency-inverse document frequency matrix better fit this type of text to give greater significance to "rarer" words (although both gave good results). Once receiving the TF-IDF, cosine_similarity() was used to find the cosine similarity of each document in relation to each other to create a similarity matrix of size n x n movie descriptions. The similarity matrix rows and columns were the same labels being that each column and row represented every movie description in subset genre. To understand the similarity matrix, the values range from 0 to 1. With this, the highest similarity is denoted by 1 and the lowest similarity between two documents is denoted by 0; therefore, the top 5 values (aside from 1, which denoted a documents similarity to itself), were extracted from the column that represented the inputted movie's document vector to be used as the 5 most similar documents based on description. [Step-by-step visual below]


Step-by-Step
n → # of documents
m → # of words in all documents
Input: "Movie Title"
Subset Full Data Frame by "Movie Title" genre
Reset Index of Subsetted Frame
Save "Movie Title" Index within Subsetted Frame
Create a Term Frequency-Inverse Document Frequency Matrix using Subsetted Frame's text
TF-IDF Matrix ex.
n x m
[0.23 0.14 0.25 … 0.14 1.43 2.34 … 0.54 0.88 1.85 … …]
Take the cosine similarity of TF-IDF to create similarity matrix
Similarity Matrix ex.
n x n
[1.0 0.1 0.2 … 0.1 1.0 0.3 … 0.5 0.8 1.0 … …]
Use saved movie position to call "movie title" column within the similarity matrix
Received Top 6 values in that column
Deleted observation with a "1" value [denoting a document's similarity to itself]
Receive indices of the 5 observations & use it to return names of most similar movies from the subsetted frame

MOVIE RECOMMENDATION - RESULTS

Overall, the recommendation function’s input is the intended movie title as a string. From this, the function calculates and prints the five most similar documents, which serve as the recommended movies one should watch. (as shown below)



```
Recommendation("Return of the Jedi")
```

```
Since you've watched Return of the Jedi, we recommend you watch The Empire Strikes Back, Star
Wars: Episode I - The Phantom Menace, Star Wars, 中國超人, or ガメラ対宇宙怪獣バイラス.
```

```
Recommendation("Finding Nemo")
```

```
Since you've watched Finding Nemo, we recommend you watch The Reef 2: High Tide, The Swan Pri
ncess: A Royal Family Tale, Shark Bait, Figaro and Frankie, or Back To The Sea.
```

```
Recommendation("The Princess Diaries")
```

```
Since you've watched The Princess Diaries, we recommend you watch Hatley High, The Prince & M
e 2: The Royal Wedding, My Father the Hero, Не покидай..., or Starci na chmelu.
```

MOVIE RECOMMENDATION - LIMITATIONS

Data frame size proved to be a large hurdle when calculating distances of a matrix. Even when subsetting for specific genres, some highly impacted genres such as “Drama” would have too many values for a distance matrix to be calculated in reasonable time. In lieu of this problem, if a single genre exceeded 10,000 values, which would begin to be time extensive, a sample of 10,000 observations of that genre from the full data frame was taken and movies were recommended from that sample rather than the full data frame. Another limitation, but relatively small, would be that genres that had fewer than 6 observations were not able to be calculated due to too few movies.