MOVIE GENRE CLASSIFICATION

CSC 575 Final Project

Abstract

In my project I utilized methods learned in class to implement a movie genre classification system that classifies movies under a given genre as well as returns the N most similar movies based off a description. We look at pre-curated Title, Genre, Description data gathered from IMDB.

Tate Teague

Tteagu4@depaul.edu

Overview

For my final project, I decided to look at text classification and recommender systems related to Movie Genres & descriptions. The goal was to build a system that given a description of a movie classify it into one of several different genre categories. This classification would be done through Term Frequency x Inverse Document Frequency matrices that would allow for callback of similar movies in the same category. This was done via the Rocchio Algorithm which we will discuss further on.

Data

The data for my project was downloaded from Kaggle, here. While the data originally seemed like a straightforward clean data set, we would come to see that was not quite the case as we would need to make numerous adjustments to remove nonsense and noise from the sample. The original table used in training and building my model consisted of 54,214 different movies, genre labels, and descriptions. There were also originally 24 different genres, with varying degrees of equivalence across their prevalence.

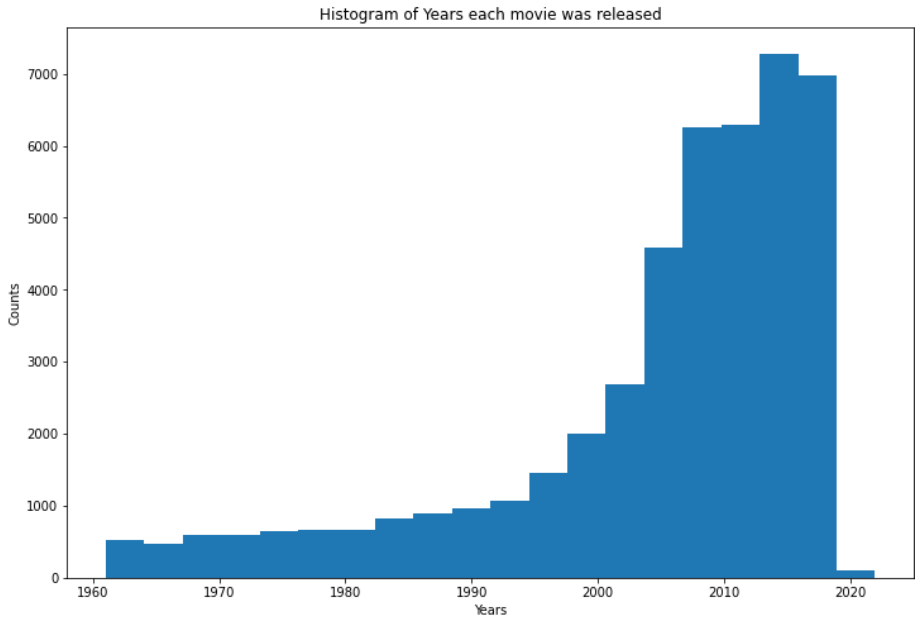
As you can see this lends to a slightly unbalanced class assortment, which would prove to be a problem when classifying the genres with fewer samples. Below are the types of data that our initial data frame contained.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 54214 entries, 0 to 54213
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   id      54214 non-null  int64
 1   title   54214 non-null  object
 2   genre   54214 non-null  category
 3   descr   54214 non-null  object
 4   label   54214 non-null  int8
dtypes: category(1), int64(1), int8(1), object(2)
memory usage: 1.3+ MB
```

drama	0.251098
documentary	0.241561
comedy	0.137363
short	0.093574
horror	0.040654
thriller	0.029347
action	0.024256
western	0.019036
reality-tv	0.016306
family	0.014461
adventure	0.014295
music	0.013484
romance	0.012395
sci-fi	0.011934
adult	0.010883
crime	0.009315
animation	0.009186
sport	0.007968
talk-show	0.007212
fantasy	0.005958
mystery	0.005884
musical	0.005109
biography	0.004888
history	0.004482
game-show	0.003578
news	0.003339
war	0.002435

drama	13613
documentary	13096
comedy	7447
short	5073
horror	2204
thriller	1591
action	1315
western	1032
reality-tv	884
family	784
adventure	775
music	731
romance	672
sci-fi	647
adult	590
crime	505
animation	498
sport	432
talk-show	391
fantasy	323
mystery	319
musical	277
biography	265
history	243
game-show	194
news	181
war	132

To gather further insight into our underlying data we could extract the rightmost 6 characters and clean them to get the year for those movies that had a valid year and plot a histogram to see the underlying distribution. My thought with this is that we might get linguistic differences from certain movie groups that were characterized differently than how they would be described in the present day.



From the distribution above we can see that this will likely not be of concern as a majority of the movies come from the mid-2000s onwards.

Method

Our first task in analyzing the data was to clean and tokenize all the words. After different iterations and examining the final results the following text cleaning methods were used:

1. Removal of Stopwords
2. Removal of non-alphabetical characters
3. Porter Stemmer stemming via NLTK package
4. Unique Movie terms & Foreign words removed were they only showed up in less than 3 movies

The above text cleaning methods yielded the most optimal results on our training set raising the accuracy by ~4%. While the first three are common practice text cleaning methods the final method is not necessarily a recommended approach. However, after struggling to increase accuracy and diving into the data I believe not only was the method justified but also significantly increased the value of rare but descriptive words that were slightly more common. The basic methodology arose from the discovery of what appeared to be either misspelled words or strings of words that were not English words. This originally produced 3,115,240 words, with 89,316 unique words to create a document by term-frequency matrix from.

```
allWords = np.asarray(dfTrain.stemmed)
fullSet = [word for sent in allWords for word in sent]
cnt = Counter(fullSet)
```
✓ 0.6s

```
print(f'# of words: {len(fullSet):,}')
```
✓ 0.1s

```
# of words: 3,115,240
```

```
print(f'# of Unique Words: {len(cnt):,}')
```
✓ 0.1s

```
# of Unique Words: 89,316
```

My first instinct was to remove words that appeared only once but this would not necessarily solve the problem as there could be multiple misspelled words or foreign words in a given genre that would cause new unseen words to always classify as that genre. Instead, I decided to look at the unique set of words per document and then looked at the words that appeared once. This ended up improving the final model

by roughly ~1% on average from ~51% accuracy to 52-53%. I experimented with the number of documents that a term must appear in and found 2 documents to produce similar results but decline from there.

```
dfTrain['reduced'] = dfTrain['stemmed'].apply(lambda a: ' '.join([word for word in a if word not in uncommonWords_byDoc]))

wordList = np.asarray(dfTrain['reduced'].apply(lambda x: x.split()))
totalWords = [word for sent in wordList for word in sent]
uniqWords = Counter(totalWords)

print(f'Total # of words: {len(totalWords):,}')
print(f'# of unique words: {len(uniqWords):,}')
```

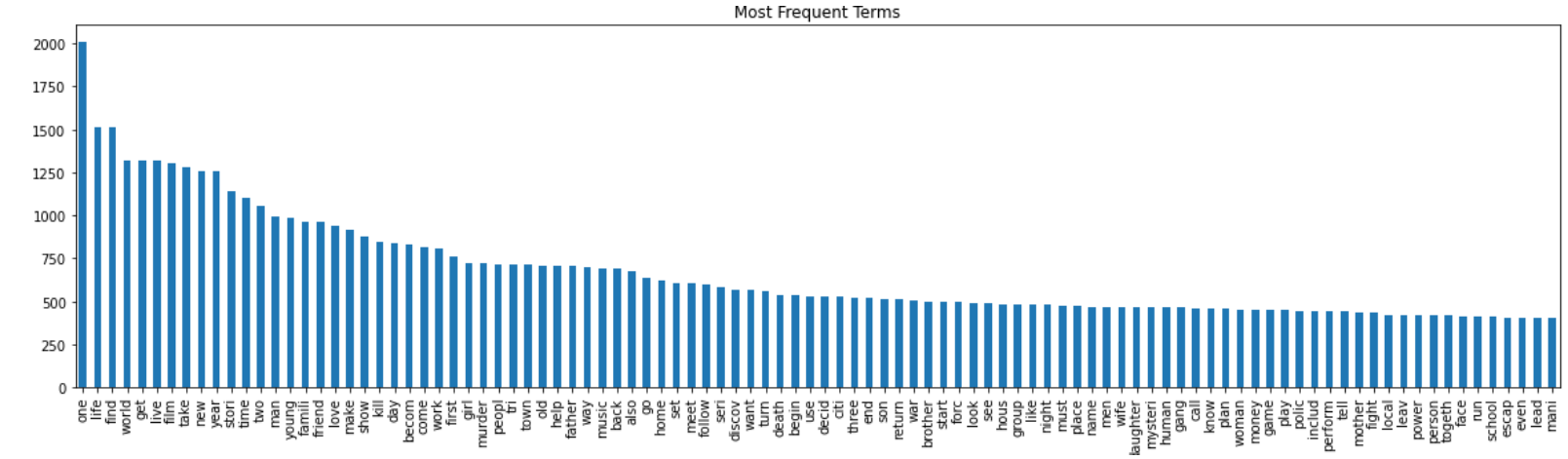
✓ 1.2s

```
Total # of words: 2,968,651
# of unique words: 72,262
```

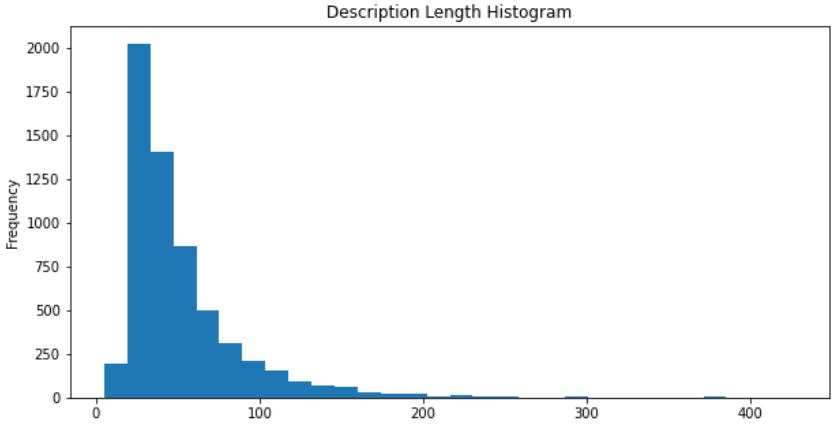
After removing the misspelled & non-English words, we were left with a corpus of 72,262 unique words and a total word count of 2,968,651. Turning this into a document-by-term frequency matrix proved to be quite a challenge as it more often than not could not allocate enough space in memory to create such a large data frame. After researching optimization techniques the most straightforward approach suggested better data type management. This meant that instead of storing everything as either a string or int64 object I could reduce space by identifying the needs of the underlying data. For example, for the term frequency matrix, there was no need to allocate int64 when I know the number could only be positive and never surpass the millions. Therefore I cast the data to uint8 which saved significant space bringing the memory requirements down by a factor of 4 from ~14.5 GB to 3.6 GB.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 54214 entries, 0 to 54213
Columns: 72242 entries, aa to zz
dtypes: uint8(72242)
memory usage: 3.6 GB
```

Below is a bar plot of the most frequent words across our corpus:



Another good feature to understand is the average length or at least the distribution of the number of words used in describing a given movie. With the removal of misspellings and stopwords, we saw the average movie length to be ~52 words and the median to be 40. Below is the underlying distribution.



The next step after creating and analyzing the term frequency matrix (doc x tf) was creating the term frequency by inverse document frequency weighting. This would help improve classification accuracy in our Rocchio model as well as provide improved feedback when looking for similar movies. As with much of the other code in this project we were able to utilize a TFxIDF function I created for one of the homework which with some small modifications worked here.

After having the core components for our classification model we could construct our Rocchio classification model. This would be constructed of three separate functions the first being a training function that calculated the prototype vectors as well as returned the unique classes. The second was a classification function that given a single test instance would compute the cosine similarity between the test instance and all other movies.

Results

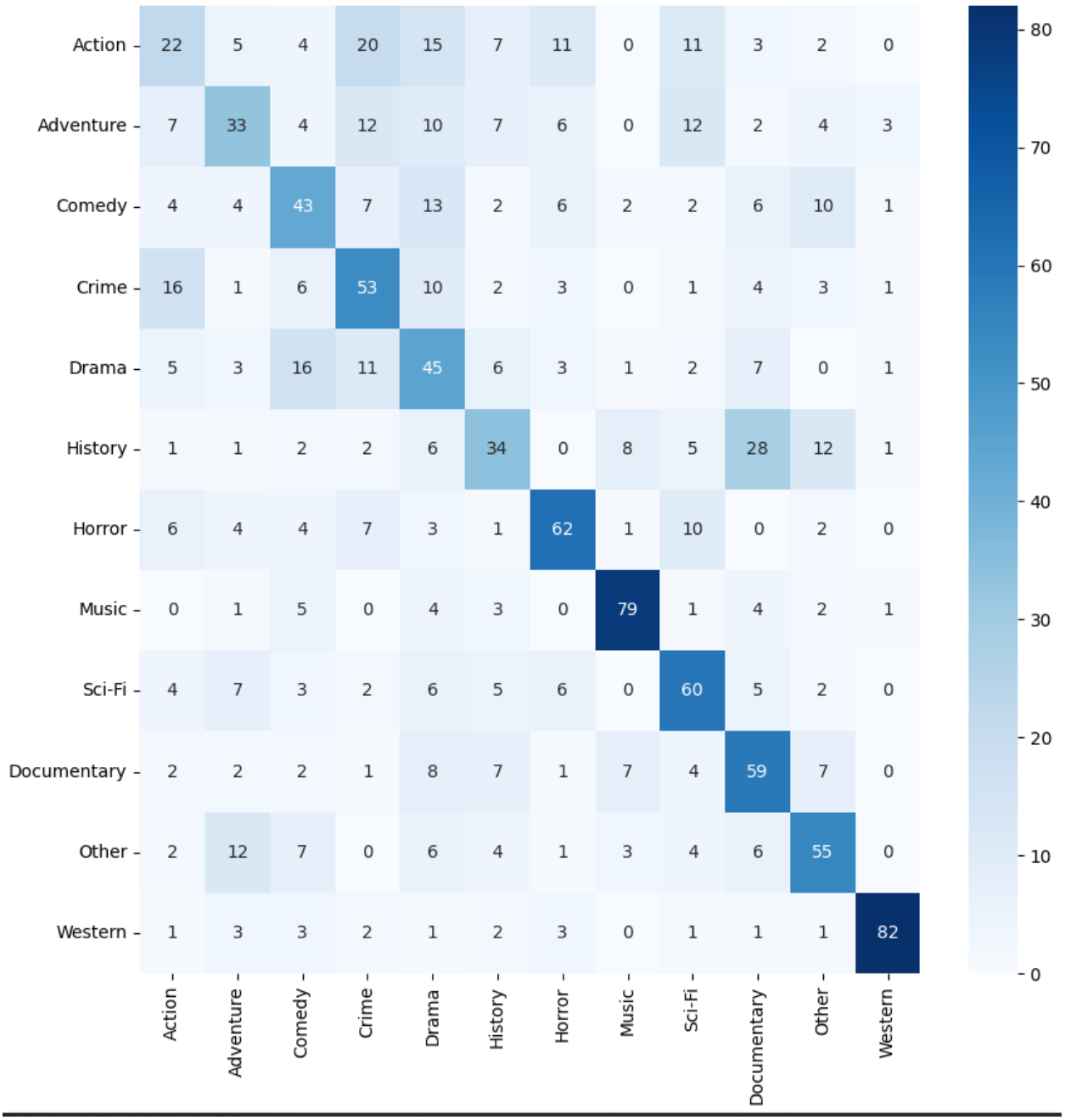
Our initial results were less than promising, after all of the care taken to clean the dataset we were still receiving an accuracy of just 42%. This caused me to add all the additional word cleaning methods described above and yet still only barely increase accuracy to ~48%. It was not until I decided to directly deal with the imbalanced classes did we begin to see better results. The best results yielded from a broadened grouping of classes and uniform distribution of each of the classes. This was done via investigating other broader commonly used movie groupings found on sites such as [Main Film Genres (filmsite.org)](). After digging in a little more I landed on the following final class groupings.

action	adventure	comedy	Crime	drama	History	horror	musical	sci-fi	Documentaries	Other	Westerns
thriller	mystery			romance	biography		music	fantasy	short	reality-TV	
				adult	war					family	
					news					animation	
										sport	
										talk-show	
										game-show	

Bringing down the class size from 28 to 12, still, after doing this there was a disparity in the number of data points for each class. This was resolved by taking the top 500 samples from each group leaving us with an ending dataset of 6,000 movies and 12 classes each with 500 samples. This resulted in a reduction of our matrices to 12,163 unique words with a total word list of 322,865. The resulting accuracy was still nowhere in the realm I was hoping for but at ~51% it was on par with much more advanced models that utilized TensorFlow and sophisticated neural networks. It does appear that some of the classes were more easily identifiable than others. Westerns continuously scored high along with Music and Horror.

Accuracy over all test set is: 52.00%

	precision	recall	f1-score	support
Action	0.36	0.21	0.26	100
Adventure	0.49	0.37	0.42	100
Comedy	0.44	0.35	0.39	100
Crime	0.51	0.62	0.56	100
Drama	0.33	0.51	0.40	100
History	0.51	0.48	0.49	100
Horror	0.59	0.61	0.60	100
Music	0.75	0.74	0.74	100
Sci-Fi	0.55	0.58	0.57	100
Documentary	0.38	0.46	0.42	100
Other	0.51	0.45	0.48	100
Western	0.86	0.86	0.86	100
accuracy			0.52	1200
macro avg	0.52	0.52	0.52	1200
weighted avg	0.52	0.52	0.52	1200



Aside from merely classifying the genre my implementation project also would provide recommendations of similar movies using cosine similarity to the movies in the document by term frequency matrix. While the cosine similarity scores were fairly low we could see that for some random movies and descriptions it gave interesting recommendations that I would be curious now to watch!.

```
Enter 1 for Test Example and 0 to enter your own movie. 999 to exit:
1
Title:
 Gunfight at La Mesa (2010)

Description:
 Tate Noble returns to the town of his youth where as a boy his parents were murdered. His childhood friend Samuel, now the sheriff of La Mesa knows who is re
sponsible, and Tate's arrival sparks hostility between Samuel and his father Judge Carter. As the mystery unravels, Tate and Samuel enlist help from an unlike
ly source, the mob, in order to bring to justice the man ultimately responsible, the evil Harcourt Simms.

This movie seems like a Western
                                  title       genre                                                descr
506           "Schlock Shock Radio" (2013)      horror  As ex bandit  Shaw  SCOTT GLEINE   travels ab...
2229                   Tempo Perso (2001)      comedy  Samuele  Nicola  Walter and Magda  are four f...
1929              "Le grand secret" (1989)    thriller  Als ihr Geliebter ebenso wie ein indischer Kr...
4665               The Iron Sheriff (1957)     western  In       in a small town in South Dakota  a s...
280    Karl Weschke: Myths of a Life (2001)  documentary  A film portrait about the moving life of Germ...
Similarity scores: [[0.17163086]
 [0.14562988]
 [0.12561035]
 [0.11669922]
 [0.11444092]]
```

The interactive element to this code gives users the chance to play with different movies and descriptions and get recommendations as well as suggestions as to the genre type.

```
Enter 1 for Test Example and 0 to enter your own movie. 999 to exit:
0
Movie Title:THe Mummys Kiss
Please input a description of a movie:A beautiful woman, named Zita, who has many issues, is aging and seeks an Egyptian goddess' help. After pledging her all
egiance, Zita is instructed to kiss the mummy remains of the evil sorceress, Hor Shep Sut. This awakens the mummy, and the murders begin.
```

```
This movie seems like a Horror
                                          title        genre                                                 descr
687    Piet Piraat en de mysterieuze mummie (2010)       family   Piet s crew is hired by aunt Cleo to return a...
3575                            Margaritka (1961)       family   The school year Margaritka is ill  Her Daddy ...
823              Mystery of the Lost Mummy (2000)   documentary   Information collected about pharaohs  mummies...
3147                      The Cat Creature (1973)       horror   When a rich man dies  some items from a colle...
4020                      Curse of Bigfoot (1975)       horror   A group of high school students on an archaeo...
Similarity scores: [[0.35375977]
 [0.33984375]
 [0.23608398]
 [0.22546387]
 [0.17858887]]
```

While it was not the most accurate system it was extremely interesting that the responses and recommendations seemed genuinely intriguing and not all that far off except for maybe some edge cases where a family movie would constantly be recommended to Horror fans.

With more time I think it would be interesting to investigate some other class balancing techniques such as SMOTE or other over/under-sampling techniques to increase total data size. The other method for improvement could be rethinking which similarity algorithm was used, instead of cosine, another method may have had increased results, though literature tends to recommend cosine similarities for these types of classifications.

Sources:

Genre Classification Dataset IMDb | Kaggle

Main Film Genres (filmsite.org)

A Guide to the Basic Film Genres (and How to Use Them) (premiumbeat.com)

Advanced Pandas: Optimize speed and memory | by Robbert van der Gugten | bigdatarepublic | Medium

4 Methods to Optimize Python Code for Data Science (analyticsvidhya.com)