Movie-genre Classifier

Park Chan Ho (3035164283)
Son Byeong Woo (2012599510)
Park Ga Youn (3035120079)
Hamza Siddiqui (3035243106)

Abstract

Given the dataset consisting 45,000 samples of movie posters, description and genre, our team aimed to carry out an multi-modal analysis of plot and poster of movie to classify genre. With the help of text embedding methodologies, neural network model, we were able to achieve around 60 % accuracy in predicting the genre of a movie when given a poster and text description of plot.

1. Introduction

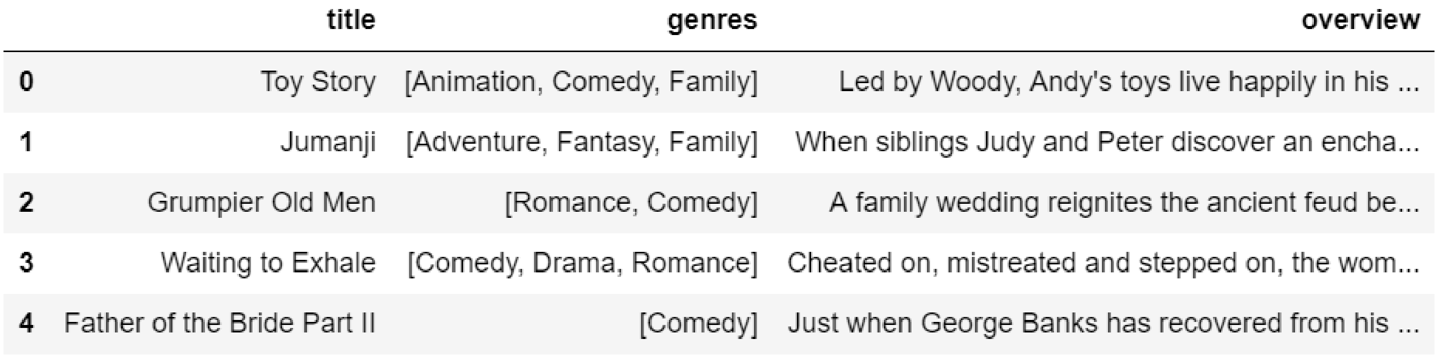
Following is the flow of our project:

1. Data preprocessing / cleaning
    a. Text preprocessing (Removal of stopwords, lemmatization)
    b. One hot encoding of genre labels
2. Text Analysis using Word2vec/ Doc2vec
    a. Creating model with our dataset
    b. Making use of pre-trained model
3. Multi-layer Perceptron
    a. Structure of MLP
    b. Hyperparameter tuning
    c. Adjustments based on confusion matrix
4. Poster Analysis using CNN (Convolutional Neural Network)
    a. Structure of CNN
    b. Accuracy
    c. Improvements
5. Combining multiple models
    a. Ensemble learning methods
    b. Evaluation of accuracy and confusion matrix

2. Data Exploration

Our main motivation was to utilize more than 1 type of modality as our input data and investigate whether this would produce a more effective model. We then decided to use text and images as our input data because these data types are easier to extract features.

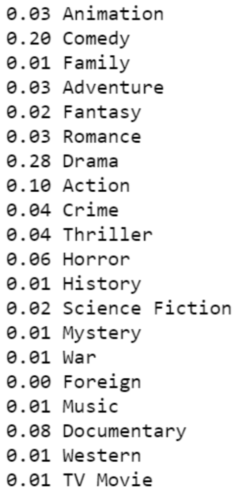
We were able to scrape 45,000 number of movies' plot and poster. The scraped data included the title, genre and the overview of each movie.



	title	genres	overview
0	Toy Story	[Animation, Comedy, Family]	Led by Woody, Andy's toys live happily in his ...
1	Jumanji	[Adventure, Fantasy, Family]	When siblings Judy and Peter discover an encha...
2	Grumpier Old Men	[Romance, Comedy]	A family wedding reignites the ancient feud be...
3	Waiting to Exhale	[Comedy, Drama, Romance]	Cheated on, mistreated and stepped on, the wom...
4	Father of the Bride Part II	[Comedy]	Just when George Banks has recovered from his ...

Usually, a movie is not solely restricted to a single genre but a combination of different genres for example, “romantic comedy”, “horror and thriller”, or “action comedy”. There were in average 2~3 number of movies in each genre. For categorical variables with no ordinal relationship, integer encoding is not enough, and using only integer encoding may result in poor performance or unexpected results. Hence, we utilized one-hot encoding to perform “binarization” of the categories. For our purpose, only the first (main) genre from the dataset was extracted to transform the data into a vector format (one-hot encoding).

There were in total 32 genres existing in our database. We narrowed our scope down into 20 genres after excluding genres with low sample number. After removing the minor labels the composition of genres looked as such:

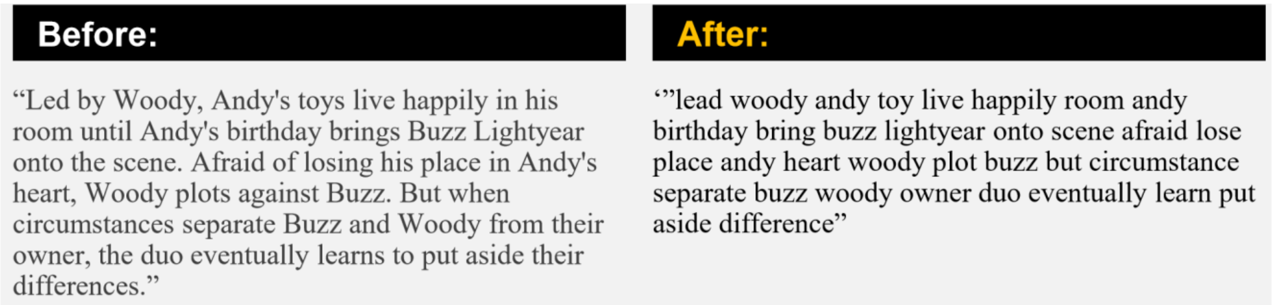


```
0.03 Animation
0.20 Comedy
0.01 Family
0.03 Adventure
0.02 Fantasy
0.03 Romance
0.28 Drama
0.10 Action
0.04 Crime
0.04 Thriller
0.06 Horror
0.01 History
0.02 Science Fiction
0.01 Mystery
0.01 War
0.00 Foreign
0.01 Music
0.08 Documentary
0.01 Western
0.01 TV Movie
```

Furthermore, The number of samples in each genre were not equal. In fact, the labels with the largest number of samples were comedy and drama, contributing to 20 percent and 28% respectively. A common issue in data mining is the size of the data set. If we use random sampling, then there is a chance that the training or the test data set may not be representative of the overall data set. In other words, from our data set of 45,000 movies and 20 generes, it is likely that one of these 20 generes is not represented in the validation test data set. This problem would lead to skewed results, hence, we applied stratification method so that each genre is correctly represented in both the training and testing data sets. Through stratification method, we tried to guarantee a correct distribution of each genre among the training and validation data sets, thereby avoiding overfitting and gaining accurate validation accuracy.

To make full use of our description data, we had to go through the process of cleaning unnecessary words and unify words of the same meaning; each respectively are referred as removing stopwords and lemmatization.

With the help of nltk text data processing toolkit, we were able to convert descriptions in such a way:



Before:

“Led by Woody, Andy's toys live happily in his room until Andy's birthday brings Buzz Lightyear onto the scene. Afraid of losing his place in Andy's heart, Woody plots against Buzz. But when circumstances separate Buzz and Woody from their owner, the duo eventually learns to put aside their differences.”

After:

‘”lead woody andy toy live happily room andy birthday bring buzz lightyear onto scene afraid lose place andy heart woody plot buzz but circumstance separate buzz woody owner duo eventually learn put aside difference”

<Cleaning description of Toy Story>

Note that the words such as “in, his, of , ‘s, to’ were all removed and verbs such as ‘led’ is converted to ‘lead’ and nouns such as ‘toys’ are converted into ‘toy’.

After employing such methods to clean our dataset, we have splitted our dataset into test data(0.15) and train data(0.85) employing the method of stratification; stratification is to keep the ratio of labels for test and train data to be consistent.

3. Machine Learning Methods

In order to convert the string description data into a quantifiable form, we have made use of a Word2Vec and Doc2Vec which are word/text embedding models developed by Google. Initially we have developed our own model based on our data. However in order to boost our performance we have found existing pre-trained models trained by corpus consisting of Google news and wikipedia. This has increased our performance by 5 percent. When looking at each text as a bag-of-words, we attempted to take average vector value of words in a text and also tried using Doc2vec to embed text based on pre-trained model. The results were as follows:

Methodology	Accuracy
Average word vector embedding method without pretrained-model	~0.43
Average word vector embedding method with pretrained-model	~0.49
Doc2vec embedding without pretrained-model	~0.41
Doc2vec embedding with pretrained-model	~0.46

In order to check the accuracy of the embedding, we have checked the cosine similarity between descriptions to see whether our embedding was intuitively meaningful. Interestingly the description that was closest in distance to movie “Toy Story”’s description was “Tin Toy”’s description (similarity ~0.8) and most different from the description of a war movie(similarity ~ 0.2).

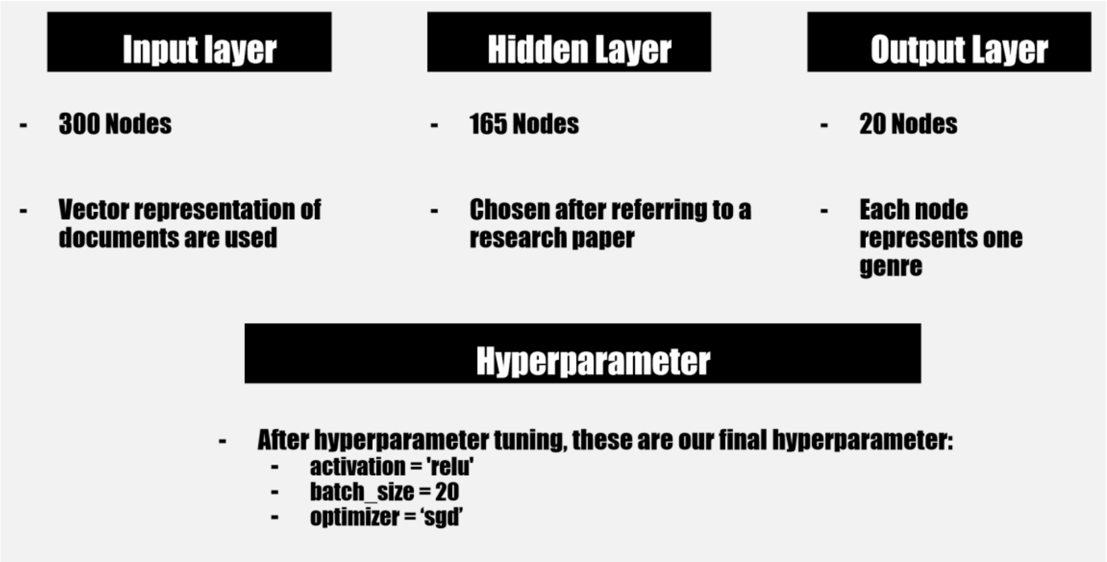
Below are the short description of each movie:

> Toy Story: “Led by Woody, Andy's toys live happily in his room until Andy's birthday brings Buzz Lightyear onto the scene....
>
> Tin Toy: “Babies are hardly monster-like, unless you're a toy. After escaping a drooling baby, Tinny realizes that he wants to be played with after all....”
>
> War movie:“Three stories of the supernatural are recounted in this anthology. Rick, an Aboriginal boy living near a swamp on Bribie Island, is haunted by an American soldier who drowned in quicksand…”

For the first two description we infer that the keywords such as ‘toy’ and ‘play’ may have affected the result to be closer while the discrepancy between words such as ‘supernatural’ and ‘soldier’ are what makes the descriptions to be distant.

Eventually we made use of average word embedding using pre-trained model to embed/vectorize the description data.

4. Result of Model



Input layer

- **300 Nodes**
- **Vector representation of documents are used**

Hidden Layer

- **165 Nodes**
- **Chosen after referring to a research paper**

Output Layer

- **20 Nodes**
- **Each node represents one genre**

Hyperparameter

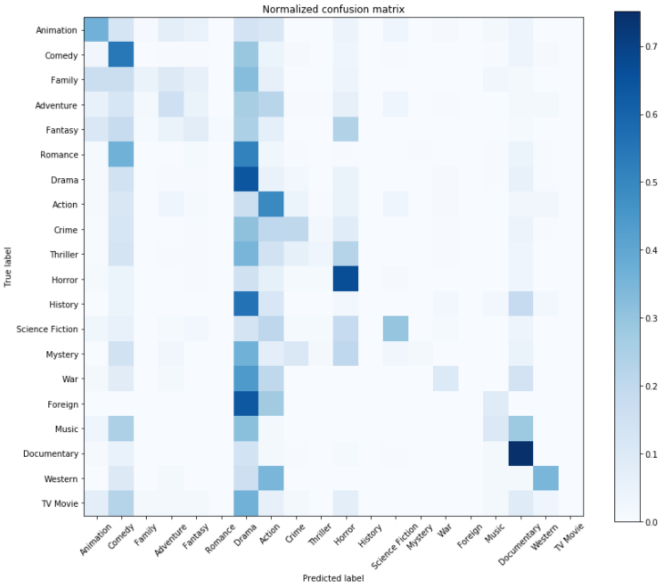
- **After hyperparameter tuning, these are our final hyperparameter:**
  - **activation = 'relu'**
  - **batch_size = 20**
  - **optimizer = 'sgd'**

After trying out multiple hyper-parameter setting and hidden layer options, we were able to figure that the most effective model in predicting movie genre is as above.

First Neural Network

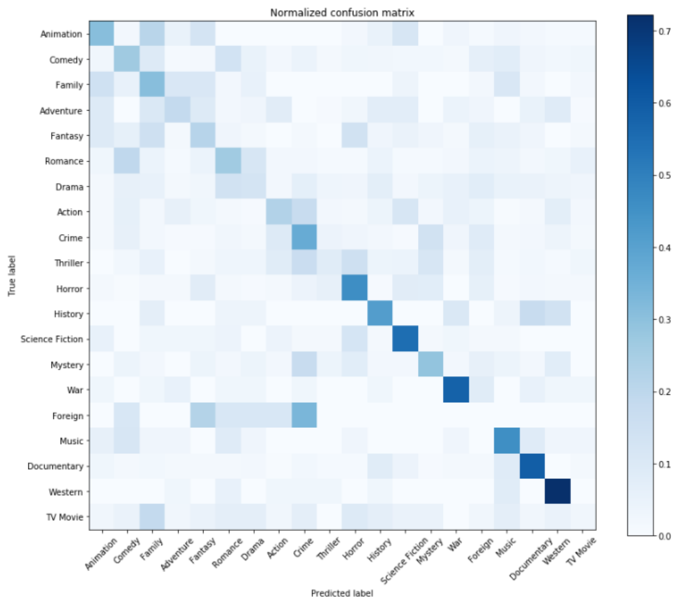
The first neural network, which is the most vanilla version as we inputted the document vector directly without any other manipulation, ironically displayed the highest accuracy of around 45%.

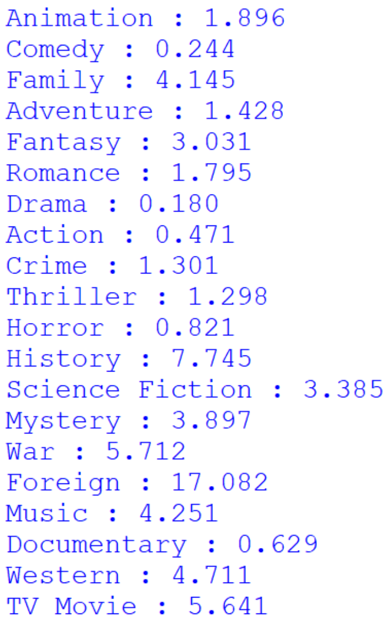
However, there is a problem with this model as there is an overfit to drama as shown by the clear vertical line in the confusion matrix above the drama label. This phenomenon is caused because the number of drama genre in the dataset is the highest at 28% so the model is trained to predict the genre as drama for more times than required.



Second Neural Network

To solve this problem, we applied class weights and gave label penalties to major labels on our second neural network. Although the accuracy dropped to about 35%, in terms of results, there is a less overfitting to drama and is more even as shown by the clearer diagonal line in the confusion matrix. The clearer a diagonal in the confusion matrix is, the better the results. The class weight we have applied are also attached next to the confusion matrix:





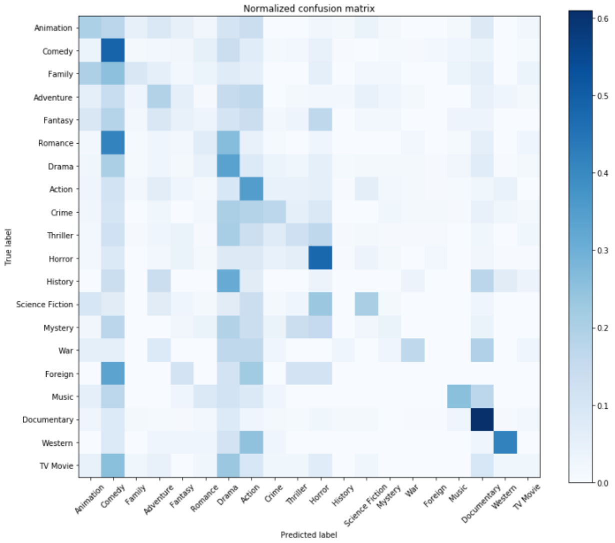
```
Animation : 1.896
Comedy : 0.244
Family : 4.145
Adventure : 1.428
Fantasy : 3.031
Romance : 1.795
Drama : 0.180
Action : 0.471
Crime : 1.301
Thriller : 1.298
Horror : 0.821
History : 7.745
Science Fiction : 3.385
Mystery : 3.897
War : 5.712
Foreign : 17.082
Music : 4.251
Documentary : 0.629
Western : 4.711
TV Movie : 5.641
```

The metric we have used to calculate the class weight was:

*(number of sample) / (number of classes * occurrence of label)*

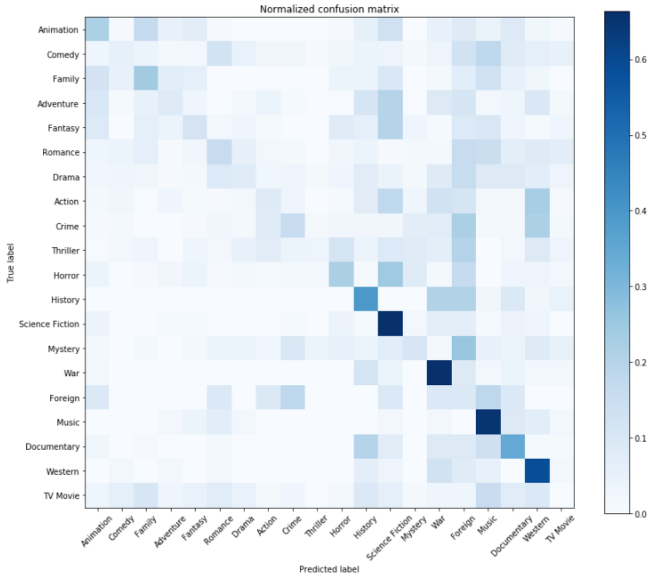
Third Neural Network

The manipulation to our model for the third neural network was to use oversampling using the SMOTE (Synthetic Minority Over-sampling Technique). It essentially makes synthetic data. In other words, this technique changes the ratio of every label to 5% because we have a total of 20 genres. So in this trial, every label has an equal chance to be selected.



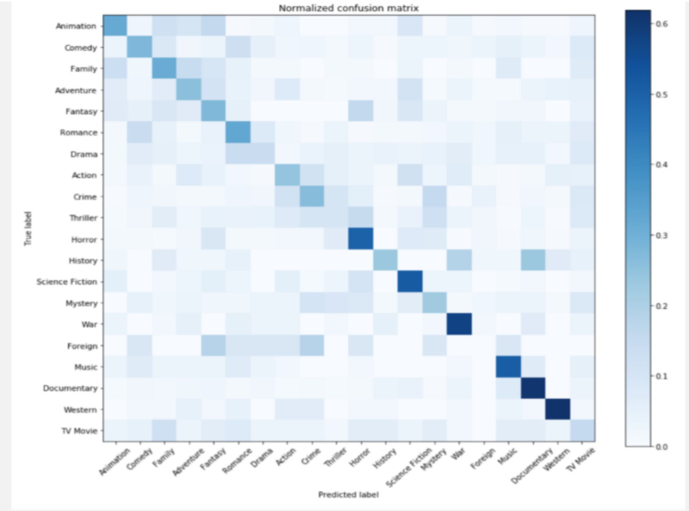
Fourth Neural Network

On our fourth neural network, we utilized the method of undersampling. The technique of undersampling uses the least common label. This resulted in the higher accuracy of predicting minor labels.



Final Model

Our final model made use of SMOTE model and class-weight model's average softmax output. The two predictions were added and averaged to give a final prediction. And the result of such method had a clear improvement in both minor and major label prediction.



Processing Movie Posters via CNN

The classification of movie posters to each of a set of genres is a task of considerable complexity: models chosen need to be capable of distinguishing and identifying the defining poster characteristics associated with a genre; this involves the detection of complicated patterns and interplay between a large number of features (278x185x3 in number). Complexity of this scale stipulates the use of neural networks. Convolutional Neural Network (CNN) models are deemed a standard for an image classification task of this nature and, thus, this was the model ultimately chosen for the poster classification task.

Data Scraping and Manipulation:

Using web addresses provided by the same 45,000 movie dataset, posters were downloaded in RGB form in the jpg image format . The images were of varying sizes, and were thus scaled to a standard resolution of 278x185x3 (the 3 corresponds to each of the 3 color channels associated with the RGB format) using image interpolation. The images were converted into numpy arrays, allowing for a representation that is more conducive to conventional machine learning models.

Structure of the CNN model:

Following multiple adjustments to both the neural networks architecture and the hyperparameters associated, a CNN model with 7 convolutional layers was chosen, with each of the layers being succeeded by an intermediary pooling layer; the function of such pooling layers is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network, thereby reducing the potential impact due to overfitting.

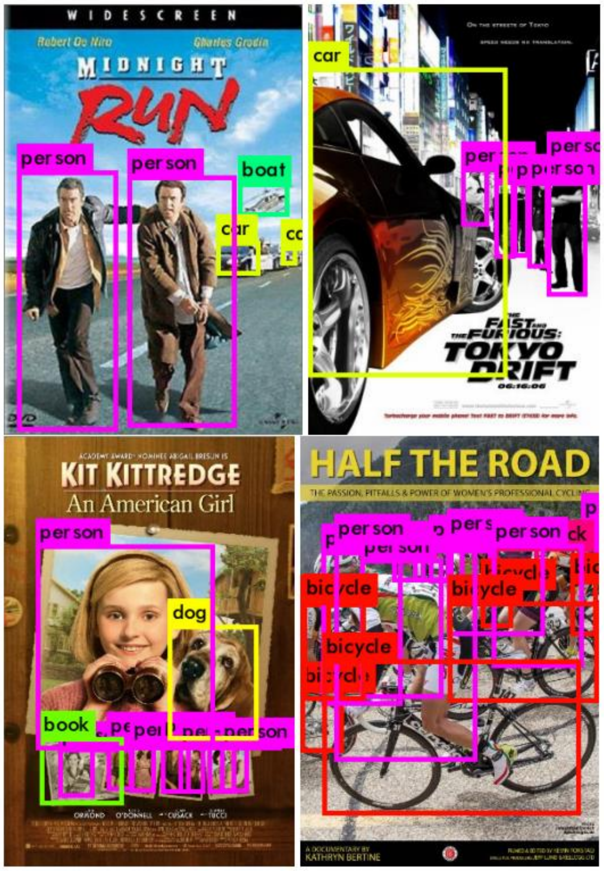
Evaluation:

Unfortunately, the model yielded a rather unsatisfactory accuracy of approximately 10%. After numerous unsuccessful attempts at boosting this accuracy, this model was ultimately discarded from consideration in the overall movie genre classifier proposed by this project.

Problems faced and suggested solutions:

The failure of the model could possibly be attributed to the fact that a large proportion of the area of the posters have information that is of little use for distinguishing between genres: for example, considerable space is generally occupied by uniform backgrounds not containing any objects of potential importance, such as persons or cars. Therefore, there is an overabundance of features, and high dimensionality, that leads to substandard results.

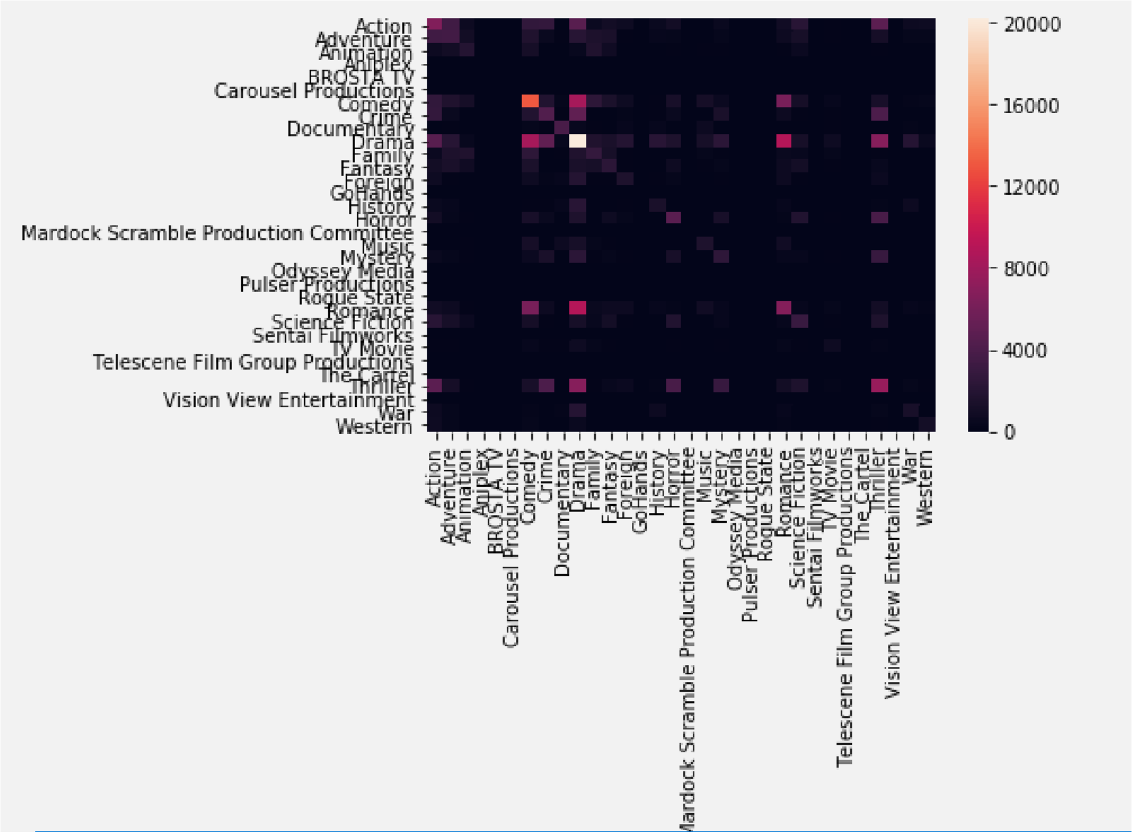
Furthermore, it is likely that the model fails to generalise effectively since the training data possibly fails to provide varying perspectives of the same objects. For example, the model might successfully classify a poster containing a car displayed in a certain angle (similar to one encountered in the training dataset) correctly, but fail to accurately classify a poster containing a similar car when encountered in an unfamiliar perspective.



A possible solution to the first problem is employing advanced feature engineering methods: these could take the form of acute feature detection methods that are capable of isolating and then broadly classifying objects contained in an image (such as people, cars, airplanes, books etc.). This would considerably reduce the dimensionality of the task, by using the types of objects in the poster, and the frequency of said object types as features (as opposed to the pixels in the entire area of the poster). Identification of the color characteristics of the poster, such as the mean color and the prominent clusters of colors present are other possible acute feature detection strategies.

The second problem could be alleviated by employing image generation methods: the dataset could be augmented by creating new images via rotation, mirroring etc. of the images in the original dataset. This would improve the robustness of the model, and would enable it to perform better with posters containing objects encountered in unfamiliar perspectives

EXPLANATION OF NEW METRIC ALONG WITH HEAT MAP



In an attempt to refine the prediction of the model, an exploratory data analysis regarding all pair combinations of genres occurring together in the dataset, was carried out. A heat map summarising the frequency of all combinations of pairs is given above. As can be observed from the graph, there is considerable simultaneous incidence of genres in the dataset (an example being Drama/Comedy and Romance/Drama).

In light of this insight, it is misguided to examine these genres in isolation, as we have done thus far: the first provided genre in the genre list for each movie in the dataset is the one that most aptly describes the movie, and this was, thus, used for both the training and the evaluation of the model. The current metric for accuracy involves comparing the output of the model against this single representative genre from each movie

Therefore, on account of the significant interactions between the genres, a better metric for accuracy would be to compare the prediction result against the entire list of genres associated with the particular movie in question, and to consider it a positive for the model if the predicted genre successfully matches up with even a single element present in this list. This modification of the evaluation methods led to an accuracy of approximately 60%. This is an encouraging result, given that the only input to the model is a brief text description of a movie, and the output is one of 20 genres.

References

Brokmeier, P. (2018). An Overview of Categorical Input Handling for Neural Networks. Retrieved from https://towardsdatascience.com/an-overview-of-categorical-input-handling-for-neural-networks-c172ba552dee

Brownlee, J. (2017). Why One-Hot Encode Data in Machine Learning?. Retrieved from https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/

Nielsen, M. (2015). Neural Networks and Deep Learning. Retrieved from http://neuralnetworksanddeeplearning.com/chap1.html

Vasudev, R. (2018). What is One Hot Encoding? Why And When do you have to use it?. Retrieved from https://hackernoon.com/what-is-one-hot-encoding-why-and-when-do-you-have-to-use-it-e3c6186d008f

Waskom, M. (2016). seaborn.heatmap — seaborn 0.9.0 documentation. Retrieved from https://seaborn.pydata.org/generated/seaborn.heatmap.html