**Movie Genre Classification by Its Poster and Overview**

Nattapat Juthaprachakul

Siyu Wu

Rui Wang

Yiyan Lan

**Introduction**

Film industry has grown rapidly in recent years. More and more movies have been released. As a result, competition has grown stronger and stronger. To attract movie goers, the film industry relies upon clever marketing campaigns. A movie poster and overview are undeniably good marketing materials that can convey theme and genre to make movie appealing to customers as much as possible. Our project is about a multilabel-multiclass classification. For our project, we divide our tasks into three parts. The first part is movie poster that has an input in form of a pixel of coloured image of a movie poster to our algorithm while the second part is an input of texts in form of movie overviews. The last part is to combine/ensemble image and text models together. The output from our models is a list of genres that classify the movie. We apply algorithms like Random Forest and deep learning models like CNN and LSTM to learn the features of the movies to make a prediction.

**Related works**

There were attempts at the models that predict a movie’s genre using its posters. A study conducted by Gabriel Barney and Kris Kaya [1] used various machine learning models including ResNet34 and KNN. Their results indicated that KNN could reach the at-least-one-match accuracy of 35.43% whereas ResNet34 could achieve a 38.26% of that accuracy. The accuracy they achieve was relatively low because classification of movie genres based on their posters alone is hard.

There were a few experiments concentrating on predicting a movie’s genre by its name and overview as most of the experiments mainly focused on sentiment analysis. A study conducted by Hadi Pouransari and Saman Ghili [2] used several approaches like random forest and logistic regression to predict the sentiment, which is binary, from movie reviews. They achieved an 84.35% accuracy for Random Forest and an 86.60% accuracy for Logistic Regression, which is a reasonably significant result.

**Dataset**

The movie dataset we used is from Kaggle which consists metadata for 45000 movies listed in the Full MovieLens Dataset. The dataset file consists of 45,466 entries. Each entry include movie id, cast, crew, movie overviews, budget, revenue, posters, release dates, languages, production companies, countries, TMDB vote counts and vote averages. The main features that we used in this project are movie posters (500 x 700 x 3 pixels) and movie overviews and the main targets are movie genres. We pre-processed the data by deleting entries with duplicated movie id (same movie names) and removing entries without listed movie genres and movie overviews. The entries that do not a valid movie poster image are also removed. We formatted every entries by deleting non-numerical and non-alphabetical characters. The final dataset after pre-processing comprises 42000 movies with the following genre distribution: Drama: 20176, Comedy:13131, Thriller:7600, Romance: 6711, Action:6582, Music:1590, Western: 1038, Adventure:3486, Mystery:2457, TV Movie: 758, Crime: 4292, Fantasy: 2304, Animation:1919, History:1394, Horror:4668, War:1322, Family:2763, Documentary:3855, Foreign:1614, and Science Fiction:3038.

A screenshot of a cell phone

Description automatically generatedA close up of text on a white background

Description automatically generated

**Features**

For image features, we wrote a script to download the original movie poster images with 700x500x3 resolution image expressed in term of RGB values and rescaled them into 100x100x3 resolution image. Hence, the inputs that were fed into our models and analysed are each single image pixel with RGB values. While for text features, we removed non-alphabetical characters and stop words like article and formatted all words into lower case. Later, we converted an array of sentences into an array of indices corresponding to words in the sentences. The genres associated with each movie are expressed in a vector of 20 lengths. Each index of vector represents each movie genre. If movie has that genre, we assign one to that index. Otherwise, the value at the index is assigned zero.

**Methods**

A close up of a logo

Description automatically generatedConvolutional Neural Network Architecture(CNN)

For pre-processing data, we added 20 columns into dataset and each column represented a genre. We used 0 or 1 (one-hot-encoding schema) to specify whether each movie belonged to that genre. After that, we loaded the movie poster, resize it and change that into array. The custom model architecture is shown in the left diagram. We used the sigmoid rather than softmax activation function for the output layer while ReLU activation function is used in other non-output layers. In addition, we used cross-entropy for our loss function and Adam for our optimizer.

Random Forest

To predict a movie’s genre by its text feature, we need to perform a preprocessing step to clean up the data. That includes putting title and overview into the same column, changing all words into lower case, removing unnecessary punctuation, and dropping all stop words. After that, we converted a cleaned sequence of words into numerical feature vectors. There are numerous approaches to accomplish this stage such as bag-of-words, word2vec and etc. The method we chose is a Tf-idf vectorizer.

After pre-processing data, we used one-versus-the-rest Random Forest classifier with 10 estimators to predict the genres.

Long-term Short-term Memory(LSTM)

For pre-processing data, we used almost the same method as we did before in training Random Forest model. Except for changing text features into numeric features, at this time we used GloVe 6B to implement the word transformation.

For LSTM model, we transformed all words in the movie title and abstract into vectors first, and later put them into LSTM model with 100 units. With the activation function of sigmoid (Dense = 20), this model outputted the probabilities of a movie belonging to each of 20 genres. Also, we used cross-entropy as the main loss function and Adam as the main optimizer in this model.

Layer (type) Output Shape Param #

=================================================================

embedding\_1 (Embedding) (None, 148, 32) 16000

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

lstm\_1 (LSTM) (None, 100) 53200

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_1 (Dense) (None, 20) 2020

=================================================================

Total params: 71,220

Trainable params: 71,220

Combining Models

To ensemble two models, one for predicting genre based on image feature and one based on text features, we assign a 0.5 weight for each model. Therefore, our combined model will be (0.5\* CNN)+ (0.5\*LSTM or Random Forest). As a result, for each movie, it had two kinds of input (poster images and movie overviews). We used two models to predict movie genre based on the probability of output for each genre. After that, we calculated the mean value of the results given by the two models. For each movie, we assign ‘1’ to that top two genres with the top highest mean probabilities of combined models and change others to ‘0’. That means, for each movie, our combined model gave it two best guesses about its genre with two highest probability.

**Results**

**A. CNN Custom Architecture with movie poster images**: We trained our CNN custom architecture on a dataset with 30000 images in the training set with 8000 in the validation set for 20 epochs and tested our model on the test set of 4000 images. We used Hamming loss, percentage of predicting at least one genre, and percentage of predicting completely all genres to evaluate our custom model. Furthermore, we analysed which movie genres that our model performed well on by calculating F1, recall and precision metric. The table below presents the genres with the highest F1 score.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Single Model: Custom CNN on movie posters | | | | |
| Genre | F1 | Recall | Precision | Number |
| Drama | 0.53 | 0.42 | 0.70 | 20170 |
| Thriller | 0.30 | 0.39 | 0.25 | 7600 |
| Comedy | 0.21 | 0.19 | 0.22 | 13131 |
| Animation | 0.20 | 0.24 | 0.17 | 1919 |
| Horror | 0.12 | 0.27 | 0.07 | 4668 |

**A close up of a map

Description automatically generated**

|  |  |  |
| --- | --- | --- |
| Custom CNN Performance | | |
| At least one match | All Match | Hamming Loss |
| 58.384% | 10.384% | 0.110 |

**B. Random Forrest with movie overview texts:** We used Random Forrest as multilabel classifier on movie overviews. We split dataset into 30000/8000/4000 train/validation/test sets. We set max\_df to 0.8 as to ignore the terms that appear more than 80% and set maximum number of trees to be 10. Again, we evaluated our model performance using percentage of predicting at least one genre, percentage of completely predicting all genres, and Hamming Loss. We also analysed the movie genres that our model performed particularly well on by calculating F1, recall and precision metric. The table below presents the genres with the highest F1 score.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Single Model: RandomForest(RF) on movie overview | | | | |
| Genre | F1 | Recall | Precision | Number |
| Comedy | 0.32 | 0.42 | 0.44 | 13131 |
| Drama | 0.28 | 0.26 | 0.31 | 20176 |
| Action | 0.14 | 0.15 | 0.14 | 6582 |
| Documentary | 0.05 | 0.10 | 0.03 | 3855 |
| Horror | 0.04 | 0.04 | 0.04 | 4668 |

|  |  |  |
| --- | --- | --- |
| Single Random Forest(RF) Performance | | |
| At least one match | All Match | Hamming Loss |
| 49.407% | 4.236% | 0.146 |

**C. LSTM with movie overview texts:** We trained our LSTM model on a dataset with 30000 movie overviews in the training set with 8000 in the validation set for 20 epochs and tested our model on the test set of 4000. We used Hamming loss, percentage of predicting at least one genre, and percentage of predicting completely all genres to evaluate our custom model. Furthermore, we analysed which movie genres that our model performed well on by calculating F1, recall and precision

metric. The table below presents the genres with the highest F1 score.

|  |  |  |
| --- | --- | --- |
| Single LSTM Performance | | |
| At least one match | All Match | Hamming Loss |
| 62.188% | 3.477% | 0.126 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Single Model: LSTM on movie overview | | | | |
| Genre | F1 | Recall | Precision | Number |
| Comedy | 0.39 | 0.25 | 1.00 | 13131 |

**D. Combined CNN + Random Forrest:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Combined Model: RF + CNN | | | | |
| Genre | F1 | Recall | Precision | Number |
| Comedy | 0.40 | 0.29 | 0.64 | 13131 |
| Drama | 0.32 | 0.29 | 0.35 | 20176 |
| Action | 0.18 | 0.21 | 0.16 | 6582 |
| Animation | 0.07 | 0.18 | 0.04 | 1919 |
| Adventure | 0.05 | 0.08 | 0.04 | 3486 |

|  |  |  |
| --- | --- | --- |
| Combined RF + CNN Performance | | |
| At least one match | All Match | Hamming Loss |
| 61.346% | 5.411% | 0.146 |

**E. Combined CNN + LSTM:**

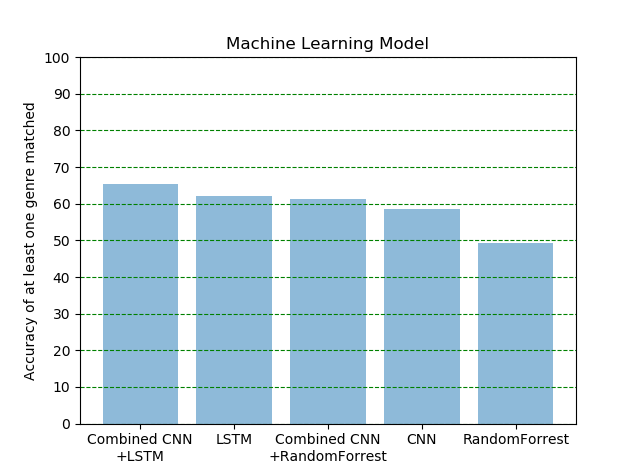
|  |  |  |
| --- | --- | --- |
| Combined LSTM + CNN Performance | | |
| At least one match | All Match | Hamming Loss |
| 65.464% | 5.364% | 0.121 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Combined Model: LSTM + CNN | | | | |
| Genre | F1 | Recall | Precision | Number |
| Comedy | 0.43 | 0.29 | 0.89 | 13131 |
| Drama | 0.26 | 0.30 | 0.23 | 20176 |
| Action | 0.18 | 0.43 | 0.11 | 6582 |
| Animation | 0.07 | 0.45 | 0.04 | 1919 |
| Documentary | 0.05 | 0.43 | 0.03 | 4668 |

**Conclusion**

In this work, we want to predict a movie’s genre based on its movie poster which is image feature, and its title and overview which are text feature. For an image part, we generated a custom CNN model and train with movie poster images. For each image, our model tried to predict movie genre based on two criteria. First criteria is if we can predict the exact match of every genre for each movie. The second criteria is whether we can predict at least one genre of each movie and we called this criteria as at-least-one-match accuracy. The at-least-one-match accuracy is the main criteria for this project. We obtained 58.38% accuracy for our custom CNN architecture, which is higher than the previous similar project conducted by Gabriel Barney and Kris Kaya[1] of 38.26%. For text feature part, we generated two models: Random Forrest and LSTM, and compared the results. LSTM model with an accuracy of 62.19% outperformed the Random Forest model with a 49.41% accuracy. Because the movie overviews were relatively short in this dataset, we considered these accuracy figures were acceptable. For the last part, we ensembled CNN model with random forest and LSTM separately. The ensembled model performed better than each of these single individual models. By combing LSTM and CNN, we got an at-least-one-matched accuracy of 65.46%, which is the highest among all of our models in our experiment.

From this experiment, we have learnt that a better performance could be generated by combing different types of features and different models. In the future, we may try to use this method to solve other machine learning problems.

However, there were some limitations in our study. Firstly, the dataset we used is relatively small and highly unbalanced. Plenty of movies in this dataset have ‘Drama’ or ‘Comedy’ labels and there are very few TV movies and western movies in this dataset. The performance of models could be different if we try that on a balanced dataset. For CNN part, we re-constructed the dataset and only kept one third of the dramas and a half of the comedies. We used the balanced dataset to train and test model. The performance of new CNN model becomes inferior with a at-least-one-matched accuracy of just 50%. Therefore, for future study, it is reasonable to train and test these models on a balanced dataset. The second limitation is when combining/ensembling models, we arbitrarily give a 0.5 weight to each of two models. Although the result we obtained shows that the combined model outperformed each of the single model, it does not mean the performance of combined model could not be further improved. Further study could explore the weights assigned to each model and could achieve a much higher accuracy.

|  |  |  |
| --- | --- | --- |
| Models | Accuracy of at least one genre matched | Accuracy of every genre matched |
| Combined CNN+LSTM | **65.464%** | 5.364% |
| LSTM | 62.188% | 3.477% |
| Combined CNN+Random  Forrest | 61.346% | 5.411% |
| Custom CNN | 58.484% | **10.384%** |
| Random Forrest | 49.407% | 4.236% |

**Reference**

[1] Gabriel Barney and Kris Kaya. Predicting Genre from movie posters.

[2] Hadi Poursansari and Saman Ghili. Deep learning for sentiment analysis of movie reviews.

**Appendix (Division of works)**

CNN model: Nattapat Juthaprachakul, Siyu Wu

LSTM model: Rui Wang, Yiyan Lan

Random forest model: Siyu Wu

Combining model: Nattapat Juthaprachakul, Siyu Wu

Report: Nattapat Juthaprachakul, Siyu Wu

Poster: Rui Wang, Yiyan Lan