Movie Genres Classification by Its Poster and Overview

Nattapat Juthaprachakul
Department of Computer Science
Simon Fraser University
Burnaby, BC V5A 1S6
nattapat_juthaprachakul@sfu.ca

Siyu Wu
Department of Computer Science
Simon Fraser University
Burnaby, BC V5A 1S6
swa246@sfu.ca

Rui Wang
Department of Computer Science
Simon Fraser University
Burnaby, BC V5A 1S6
rui_wang_13@sfu.ca

Yihan Lan
Department of Computer Science
Simon Fraser University
Burnaby, BC V5A 1S6
yihan_lan@sfu.ca

Abstract

The film industry increasing relies upon movie posters and its overviews in order to attract public attention in the hopes of generating huge profit and viewership. A genre of movie is one of the very important factor that helps movie goers decide which movies they want to spend time and money on. The project uses a poster and overview of movie to predict its genre. Therefore, our goal is to build and compare several multilabel-multiclass classification models based on classical machine learning algorithm and deeplearning. CNN model is used to train our model with an image feature while Random Forest and LSTM models are applied to train ours based on a text feature. Later, the combining models with the text and image feature are used to obtain comprehensive results. In this project, we found that the combination of LSTM and custom CNN model is reasonably successful at predicting genres with the highest at-least-one-matched accuracy is at 65.46%.

1 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 colored 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/assemble 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.

2 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.

3 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 includes 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 entry 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.

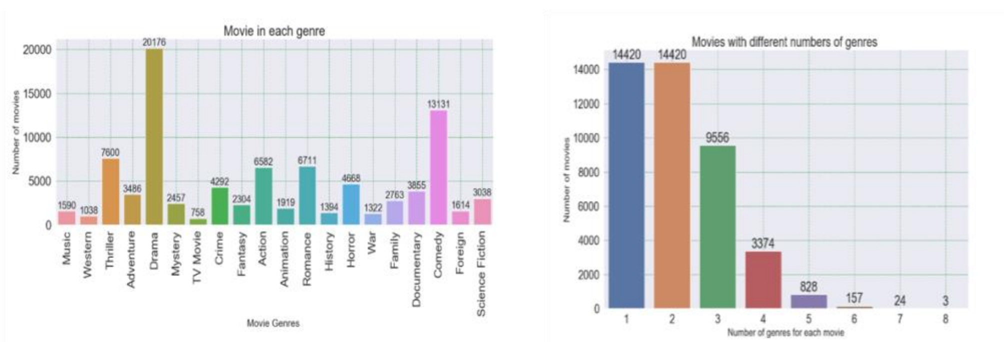

Figure 1: Movie Genre Distribution in Dataset

4 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 analyzed 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.

5 Methods

5.1 Convolutional 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.

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 100, 100, 32)	896
activation_1 (Activation)	(None, 100, 100, 32)	0
conv2d_2 (Conv2D)	(None, 98, 98, 32)	9248
activation_2 (Activation)	(None, 98, 98, 32)	0
max_pooling2d_1 (MaxPooling2	(None, 49, 49, 32)	0
dropout_1 (Dropout)	(None, 49, 49, 32)	0
conv2d_3 (Conv2D)	(None, 49, 49, 64)	18496
activation_3 (Activation)	(None, 49, 49, 64)	0
conv2d_4 (Conv2D)	(None, 47, 47, 64)	36928
activation_4 (Activation)	(None, 47, 47, 64)	0
max_pooling2d_2 (MaxPooling2	(None, 23, 23, 64)	0
dropout_2 (Dropout)	(None, 23, 23, 64)	0
flatten_1 (Flatten)	(None, 33856)	0
dense_1 (Dense)	(None, 512)	17334784
activation_5 (Activation)	(None, 512)	0
dropout_3 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 20)	10260

Total params: 17,410,612
Trainable params: 17,410,612

Figure 2: Convolution Process

5.2 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.

5.3 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

Figure 4: LSTM Process

5.4 Combining Models

To assemble 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.

6 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 analyzed 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.

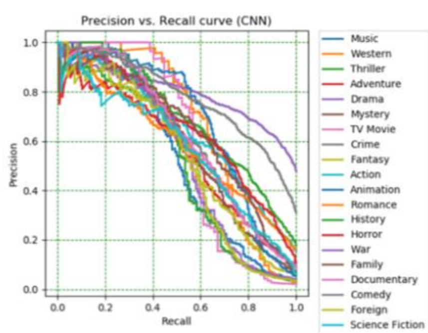

Figure 1: Sample Figure Caption

Table 1: Performance of CNN Model

Custom CNN Performance		
At Least One Match	All Match	Hamming Loss
58.384%	10.384%	0.110

Table 2: Performance of CNN Model on different genres

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

B. Random Forest with movie overview texts: We used Random Forest 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 analyzed 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.

Table 3: Performance of Random Forest Model

Single Random Forest (RF) Performance		
At Least One Match	All Match	Hamming Loss
49.407%	4.236%	0.146

Table 4: Performance of Random Forrest Model on different genres

Single Model: Random Forest (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

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 analyzed 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.

Table 5: Performance of LSTM Model

Single LSTM Performance		
At Least One Match	All Match	Hamming Loss
62.188%	3.477%	0.126

Table 6: Performance of LSTM Model on different genres

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

D. Combined CNN + Random Forest:

Table 7: Performance of RF+CNN Model

Combined RF + CNN Performance		
At Least One Match	All Match	Hamming Loss
61.346%	5.411%	0.146

Table 8: Performance of RF+CNN Model on different genres

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

E. Combined CNN + LSTM:

Table 9: Performance of LSTM+CNN Model

Combined LSTM + CNN Performance		
At Least One Match	All Match	Hamming Loss
65.464%	5.364%	0.121

Table 10: Performance of RF+CNN Model on different genres

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

7 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 Forest 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 assembled CNN model with random forest and LSTM separately. The assembled model performed better than each of these single individual models. By combining 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 combining 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/assembling 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.

Table 11: Performance Comparison of different Models

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 Forest	61.346%	5.411%
Custom CNN	58.484%	**10.384%**
Random Forest	49.407%	4.236%

References

[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.

Contribution

CNN model: Nattapat Juthaprachakul, Siyu Wu

LSTM model: Rui Wang, Yihan Lan

Random forest model: Siyu Wu

Combining model: Nattapat Juthaprachakul, Siyu Wu

Report: Nattapat Juthaprachakul, Siyu Wu, Rui Wang, Yihan Lan

Poster: Rui Wang, Yihan Lan