CS109b Project: Machine Learning for Movie Genre Prediction

By Danqing Wang, Wenshan Zheng, Zecai Liang

Have you ever wondered what makes a movie a *romance* movie when we search for popular romance movie on IMDb? Kiss scenes? Dating plots? Princess and princes? While these are some good guesses, it is difficult to pin down the exact criterion that defines a movie as belong to a particular genre. Movie genres are like pornography: it's hard to define, but you know it when you see it [1]. In this study, we try to tackle this problem from a machine learning perspective, taking advantage of the massive movie database and movie posters available on the internet, and build classical classification models and deep learning networks to train the machine to predict movie genres.

1. Exploratory data analysis

We collect our movie data from IMDb and TMDb, both of which are very popular and are considered as the most important resources about movie registries. We request 25K movies entries (randomly sampled by 'imdb_id') from both databases and directly combine their feature and genre information to form our preliminary dataset.

After combining, we have 58 features and 27 movie genres. Notice that each movie can have multiple genre labels, ranging from 1 to 7 in our data set. We first do preliminary feature cleaning by removing duplicated features such as different versions of movie 'title' and non-informative features such as 'cover url', and results in 35 features for prediction: 'title', 'director', 'distributors', 'year', 'rating' and so on.

We further explore genre-feature relationship by looking at heatmaps/histograms between genres and features. We do find that some features are strongly correlated to which genres the movie are. For example, 'Horror' and 'Thriller' movies most often have the lowest ratings, while 'Crime' and 'Drama' movies most often have the highest ratings. We also find that movies from certain actors/directors tend to be in one or a few genres. Some of these results are shown in Figure 1 below.

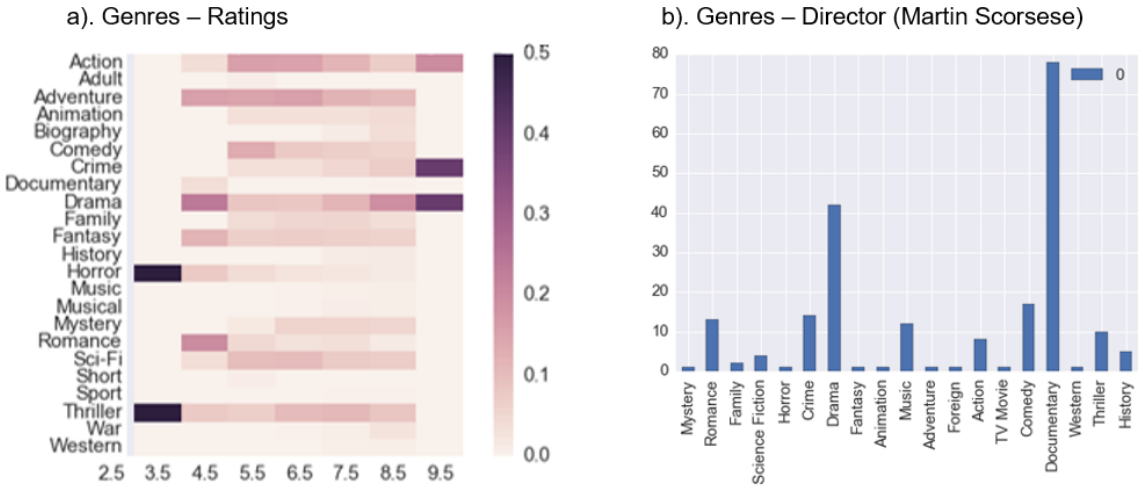

Figure 1. Exploratory data analysis of the relationship between features and genres: (a) heatmap of genres and ratings. (b) histogram of movie genres by Martin Scorsese

[1] Peter Lattman (September 27, 2007). "The Origins of Justice Stewart's 'I Know It When I See It'". *Wall Street Journal*. LawBlog at The Wall Street Journal Online. Retrieved December 31, 2014.

Staring at the distributions of the original 27 genres from the two databases (Figure 2a), we identify quite a few genres with really low movie counts. We also notice from the correlation heat map that a few genre pairs are highly correlated. Thus we decide to compensate the imbalanced nature of genres by combining highly correlated genres (Figure 2b). After grouping, we further remove genre ‘other’ which is the least common. In the end, we have in total 8 genres and each genre at least has more than 10% of movies assigned to it. Moreover, correlation between the final 8 genres is weaker after grouping (Figure 2b). Now the classification problem becomes a multi-label problem with 8 genres to predict.

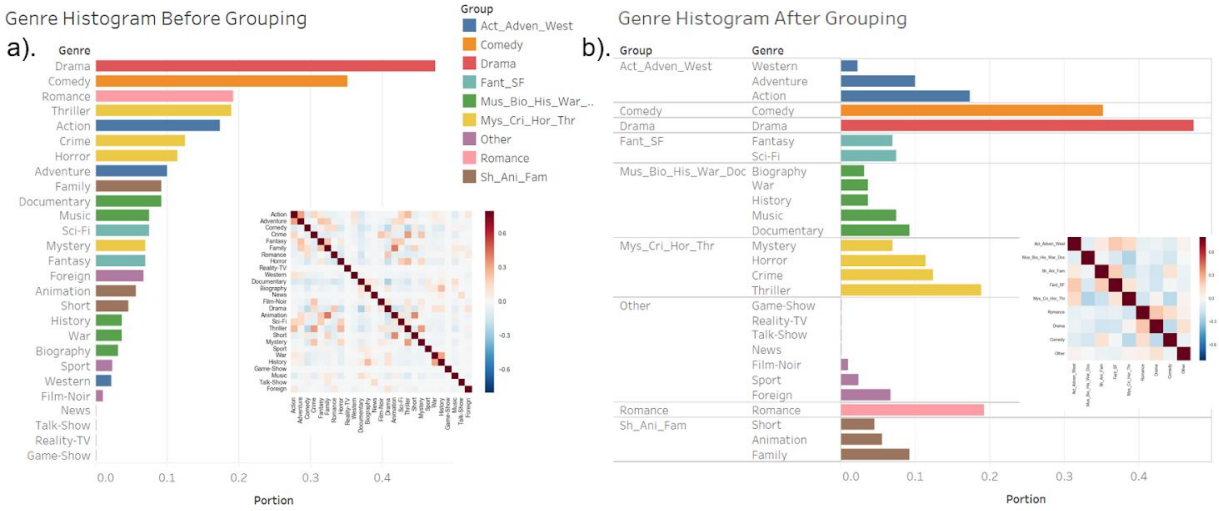

Figure 2. Genre histograms before (a) and after (b) grouping according to correlations between genres. (b) shows the final 8 genre groups that we plan to use in our analysis in the following.

2. Data pre-processing and feature engineering

From the 25,000 movie entries, we randomly sample 3000 movies as the training data, and 2000 movies as the test data. Within the total 35 different features, some variables (such as ‘directors’, ‘casts’) are a long list of names, some (such as ‘title’, ‘plot’) are paragraphs of words. So we perform some feature engineering to turn them into ready-to-use variables.

2.1 Numerical metadata

We begin by selecting the relevant numerical metadata: 'year', 'rating', 'votes', 'popularity', 'runtime'. We impute the missing values in training and test datasets with the mean value of the training data.

2.2 Dimension reduction in production-related features

We then take a look at production-related information. Since personals and companies (such as the ‘director’, ‘writer’, ‘cast’, ‘distribution company’) differ much from one movie to another, including all these information would make our feature matrix extremely sparse. Instead, we extract the most prolific director/writer from each of the 8 genres and use them as new predictor columns. For cast and production companies, we select the top 5 most representative

individuals/companies in each genre, and perform PCA. We include the top 10 principal components (PCs) that account for ~ 10% of the variation. We then use these PCs to transform the test dataset into the new basis, and prepare it for model fitting. Since these data-frames are relatively sparse, we expect some of the individuals/companies selected by the training data analysis not represented at all in the test dataset, and therefore will have columns with NULL values. We replace these NULL values with 0s. For 'animation department' and 'original music', we choose to retain information of the number of unique person IDs which representing the number of staff working there, since different movies may have either a relatively large or a small animation/music team depending on the genre, and individual names may not matter much here.

2.3 Text analysis on text-related variables

Although not without surprises, people have been guessing a movie's genre from its plot or title all the time. We therefore collect all the text-related information from the two databases and read between the lines (literally) to extract the hidden information.

We collect 'title', 'plot', 'plot_outline' from IMDb database; 'tagline', 'overview' from TMDb database; and extract the one sentence explaining the reason of the movie's MPAA rating from 'mpaa'. An exploratory text analysis on each variable alone (following the procedure in Figure3) doesn't show strong predictive power for any of the six variables, so we decide to pool all the text information for following analysis, hoping that the noise would cancel out and the text signature of each movie could jump out.

For each movie, we pool all the text together, split the sentence into collections of words, and remove the common stop words in English (such as "between", "you", "if", "when") that are rather non-informative (we used the stop-word library from the NTLK library, a powerful tool for text mining). With the resulting Bag-of-Words for each movie, we calculate the Tf–idf (term frequency–inverse document frequency) score (using TfidfVectorizer in the sklearn library), which counts the number of times a word appears in the document (weighted by the frequency of the word in the corpus). This reflect how important a word is to a document in a collection or corpus.

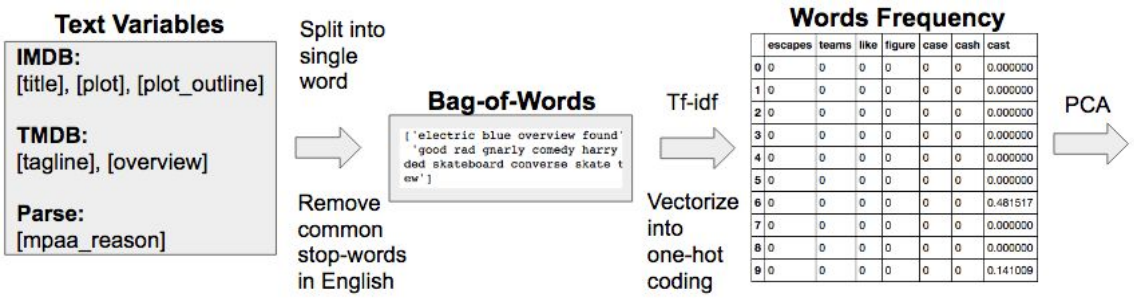

Figure 3. Flowchart of text analysis on pooled text-related variables

We then apply PCA to the word-frequency matrix of the training data, and transform the test data accordingly. The visualization of the training data (Figure 4a) shows some clustering pattern of genres when we highlight the most frequent genre in red ('Drama' is more than 40%

among the 25k movies). We then choose the top 283 PCs that could explain 60% of the variance and incorporate them as new variables (Figure 4b).

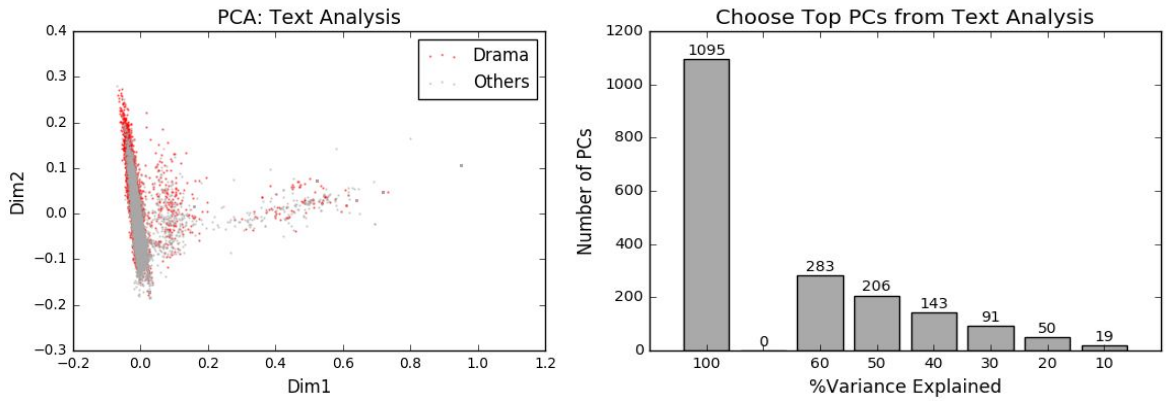

Figure 4. (a) visualizing PCA of text analysis of the training data: each point represents one movie and movies with genre ‘Drama’ are highlighted in red. (b) variance explained by top principal components

After the feature selection and engineering, we assemble a data frame of 344 variables, summarizing basic numbers, production staff and text information of each movie. Finally, we standardize our data before proceeding to building a few baseline models in the following.

3. Classical machine learning

Since the genres of a movie are not unique (for instance, a movie can be both a ‘comedy’ and a ‘drama’), we consider the genre prediction task as a multi-label problem, in which each movie can be labeled with multiple genres, as is with what we observe from the online databases.

3.1 Performance metrics

For each movie to be predicted, we use the `f1_score` function from `sklearn.metrics` package to compare the predicted value (8-long vector with 0 and 1 for each genre) and the true value. To evaluate the model, we use the averaged F1 score. We choose F1 score because we care about both the false positive and false negative mistakes. For now we are giving them the same weights, so we take the harmonic mean of precision and recall. But this can be further adjusted depending on the application of this prediction (whether it's worse to misclassify a genre or to miss a genre).

3.2 Baseline models

We start by testing some baseline models. The first models we try are KNN, decision tree, and random forest, since the sklearn package supports multi-label classification in these methods. We then proceed with LQA, QDA, SVM and logistic regression. The sklearn package do not support multilabel classification for these classifiers, we thus perform classification genre-by-genre, and combine the results after fitting all 8 genres. Without tuning these model for specific parameters, we manage to achieve a reasonable test accuracy (Table1 & Figure5). The weighted logistic regression model and the SVM model seem to have relatively high F1 score,

while KNN, random forest and others perform poorly. The SVM and RF models are computationally expensive, but have larger rooms for improvement by tuning the parameters. The logistic regression is quick and easy to interpret.

	KNN	LDA	QDA	RF	Tree	Tuned SVM	Unweighted Logistic	Weighted Logistic	Weighted SVM
f1_score	0.243856	0.579514	0.369605	0.275903	0.394552	0.608692	0.586089	0.587911	0.57734

Table 1. Table of accuracy scores of different models before tuning.

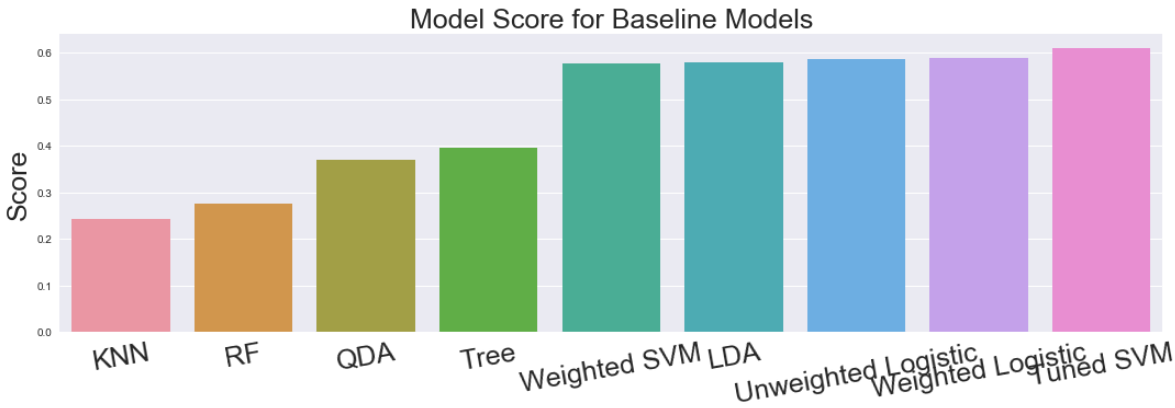

Figure 5. Accuracy score of different baseline models and tuned SVM model. The tuned SVM model has the highest accuracy of 60.9%, an improvement from the 57.7% of the untuned SVM.

3.2 Tuning SVM

We tune the SVM model by using both the linear and rbf kernels. We also tune the gamma values and accuracy compensation parameter C over the range of -7 and 7 for each genre. The best C value for each genre turn out to be 1. This indicates that the accuracy compensation is largely the same for each genre in our SVM model. We achieve an overall accuracy of of 60.9% on the test data set with the tuned SVM model.

4. Deep learning

For deep learning models, we use the same 5k movies for train and test. To prepare their posters for deep learning models, we read each poster as numpy array with dimension [224, 224, 3], corresponding to 224 x 224 RGB picture. We further center each array and concatenate them into one file for further CNN deep learning. For MLP data preparation, each numpy array is flattened before it is concatenated.

Since this is a multi-label classification problem, we use ‘sigmoid’ as activation function and compile our deep learning models with ‘binary_crossentropy’ as loss function. With this settings, we can train our model, predict one movie and get a probability vector of length 8 for all eight genres.

First we test a baseline Multi-Layer Perceptron (MLP) and optimize the model with L2 regularization and smaller learning rate. The accuracy reaches 74.4%. We further tune with

smaller batch, more nodes, more epochs and more hidden layers but the prediction accuracy doesn't improve much.

We then switch to the Convolutional Neural Networks (CNN), which is supposed to be more powerful for image processing. A pre-trained VGG16 model gives an unsatisfying accuracy of 0.67, so we decide to start from scratch and build our own CNN model. After playing with extra layers and tuning parameters, the final model with 18 hidden layers (Figure 6c) reaches accuracy of 74.4%. The loss function and accuracy from the model fit also converges smoothly (Figure 6a, 6b).

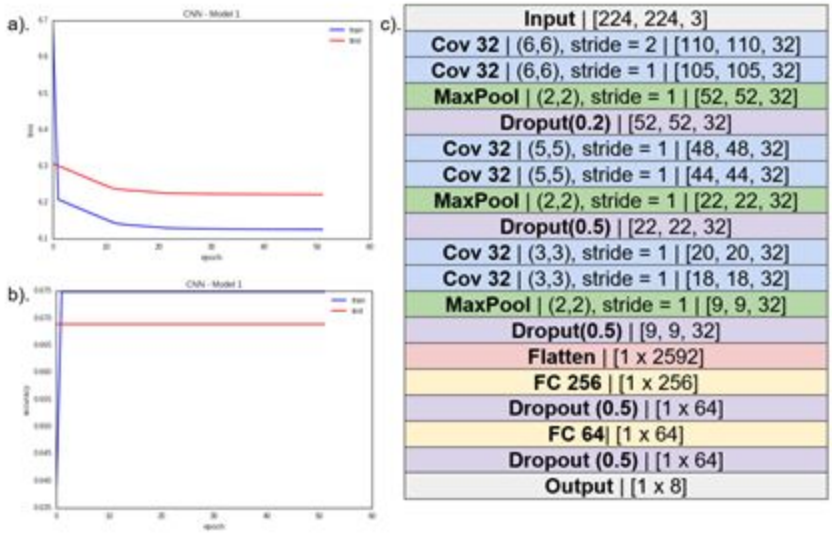

Figure 6. (a) loss plot, (b) accuracy plot and (c) model structure of final CNN model

The sparsity of the response matrix (binary coding for each of the 8 genres) is a little problematic when we use "accuracy" as metrics, since it induces prediction of mostly 0. A customized loss function similar as F1 score is more desired as it penalizes both false positives and false negatives.

4. Conclusions

We scrape 25k movie information and posters from IMDb and TMDb database, and choose 5k to assemble into training and test data after data cleaning, data re-leveling, text mining and feature engineering, and use these features to predict the 8 movie genres we re-define. A broad panel of classical machine learning models are tested and fined tuned, and the winner goes to weighted SVM with a prediction F1 score of 61%. In parallel, we feed the posters of the same 5k movies into various deep learning models (MLP, pre-trained VGG16 and customized CNN), and harvests accuracy of 74% with a fine tuned 18-layer CNN model. Future improvement could be achieved if we (1) scale up to the whole 25k or even more data, (2) implement customized loss function of F1 score to reduce false negative predictions, and (3) integrate the predictions from SVM and CNN for a mixed model.