

Final Project Report (CS109b)

Team 37 (Alexander Dubitskiy, Keenan Venuti, Timur Zambalayev)

Project repository: <https://github.com/adubitskiy/cs109b>

Link to the Google Drive folder with data:

<https://drive.google.com/open?id=0B9PSivXSSQOTQWY2X0kyUTBNOFU>

Predicting movie genres

We all love watching movies. We all have preferences with regard to genres. Some of us like comedies, some - horror movies.

But how easy or difficult is it to determine the genre of a movie by looking at other attributes of a film (the overview, main actresses, or a movie poster)? That is what we were exploring in our project.

Data and Features

Data Collection

Our task, identifying movie genres based on a self-chosen feature set, utilized two important Python dictionaries for data mining and collection. We used IMDBpy and tmdbsimple which contain methodologies to harvesting information held on IMDb and themoviedb.org, respectively. These datasets hold features such as movie overview, plot, director, actors, crew data, budget, revenue, voting information, reviews, etc. TMDB alone hosts 333,163 movies. For our learning models, we began with a 3k and 5k sub sample before ultimately using a datasets with 10k movies. This dataset became an accumulation of both the IMDb and TMDB movie datasets as they were able to complement each other where some information had not been included. For example, some movies in the TMDB dataset lacked certain revenue, budget and plot data; querying the IMDb system helped account for some discrepancies. When exploring the data, we saw the need to initially try different types of supervised models before making a decision. The datasets are incredibly diverse in potential applications. Both contain hard quantitative data like revenue, budget and average viewer rating, categorical data like actors,

directors, crew and textual data such as an overview or plot. Given the stark differences between how these different types of data will contribute to a model, we decided to focus our model's feature sets on relevant predictors that were similar to each other.

Data Cleaning

Initially, we toyed around with the datasets, cleaning them, and running small supervised learning classification to get an idea of any potential problems and the limitations of the dataset. We began with a few models based on spending, budget, ratings and quantitative reviews. Here, we began to see some of the problems when classifying movie genres. Firstly, movies fall under multiple genres. One movie may be categorized as just a Drama whereas another may be a Drama Comedy. There are a couple of approaches to this type of problem. A data scientist can focus on the genre title that best describes the movie, create hyper-classes via a combination of movie genres and perform multi-class classification, or we can modify the dataset to use multi-labels, where each movie is assigned a vector (the size of the number of all genres) with a binary value if that genre is used in describing the movie. We focused on a multi-label approach as we wanted our model to have strong applicability to the dataset and not make any assumptions. However, we did also utilize a multi-class system when experimenting with neural networks to assist in the network's learning process. After defining our `y` value, we saw the need to select our feature set based on attributes that are similar to each other. But, after creating of this classification `y` set we noticed that there was immense class imbalance in the dataset (see Figure 1). For example, genre is used to define a lot of movies. This will lead to models that over-classify the dominant class in an attempt to increase accuracy hurting applicability. We dealt with this further when selecting our models.

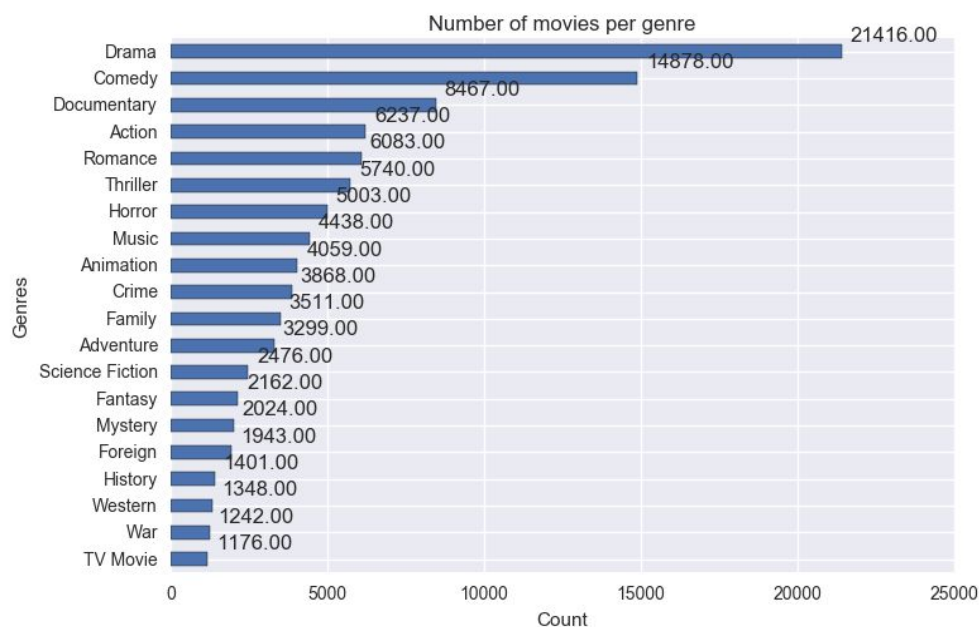


Figure 1. Number of movies per genre.

Picking Features

One of our initial feature sets focused on director, actor and crew information; this was comprised of around X features. Movies often contain the same actors playing the same role and directors almost always stay within a genre. But, we also saw predictive power in the overview/plot descriptions within the dataset. From there we implemented another model using overviews as a feature set via vectorization of the descriptions through bag of words. This process involves taking all the overviews in the dataset, eliminating stop words and creating a vector, sized at this new "bag of words". Then, each overview is represented as a vector based on which words were used. Eventually, we saw that subsetting the data based on features created accuracy weaknesses. We combined our features sets into one more complete dataset and looked to picking a supervised learning model.

Traditional Machine Learning Methods

Experimenting with Supervised Learning Models

We trained several classification models including a LDA, QDA, random forest, tree and SVM (linear and radial kernels). Ultimately, we chose a random forest and linear support vector machine with a stochastic gradient descent learning method (SGD). We found that these models were well suited for our two types of datasets (director/actor/crew and overviews). They were also different enough to properly express the data. Specifically, our SGD model cannot natively support multi-label classification. Instead, we used a one vs rest classification method within the model. This fits a different model for every class in the multi-label y set. Conversely, the random forest can natively handle multi-label data. The SGD model is based on a linear SVM, so it gains some of the qualities of that model. It can train very quickly and performed decently when compared to the random forest. The random forest takes more time as it fits specific curvatures of the train data. With the selection of these two very different models and their application to our combined dataset datasets, we focused on tuning.

Tuning

We tuned both our SGD and random forest model using an sklearn function, GridsearchCV. This takes in parameters to tune models on and finds the best combination to implement in the final model. After testing different alpha estimators through cross validation for our SGD model, we found an optimal value and created the final model. The random forest model was tuned on max features and the minimum sample leaf via grid-search. Both models were tuned based on their hamming loss score.

Baseline prediction (dummy classifiers)

To better understand our model's performance, we implemented three different dummy classifiers. Using these comparisons, we could determine if our models had applicability or were as helpful as guessing a certain way. Within our dummy classifiers, the stratified classifier is a classification algorithm that takes the training set class distribution and assigns classes to the test set based on that distribution. The uniform dummy classifier assigns classes in uniform to all observations in the testing set. Finally, the most frequent dummy classifier assigns the most frequent class to every observation. However, in this context, our classes are multi-labels.

Performance of Models

We compared accuracy across genres of both our SGD and Random Forest model against each other and the dummy classifiers. Choosing the right metric is a challenge within itself as some may be misleading while others are not relevant. For example, our classifier that assigned every observation to the most dominant class had the smallest hamming loss, but a model like this is very unhelpful. We decided to use the f-1 score as a our most important metric to understand a model's accuracy; it incorporates both precision and recall (alone these metrics can be deceiving) to give a fuller understanding of a model's strengths and weaknesses. Below is the performance of each model (see also Figure 2):

Averages - SGD Average Precision: .575 Average Recall: .544 Average f-1 score: .548 - Random Forest Average Precision:.537 Average Recall: .116 Average f-1 score: .177	 - Dummy Models Uniform Classifier Average Precision: .205 Average Recall: .502 Average f-1 score: .263 Most popular classifier Average Precision: 0.0 Average Recall: 0.0 Average f-1 score: 0.0 Stratified Classifier Average Precision: .202 Average Recall: .2 Average f-1 score: .201
--	---

Deep Learning Methods

Our traditional machine learning techniques provided impressive accuracy when compared to dummy classifiers. However, there are limitations when using these types of models. First off, they are trained and run on overview and crew/director/actor data. They make no utilization of the rather large and informative movie poster database. Secondly, with the amount of data we have on movies, there are better methods to training a classification model. Using the f-1 score as an accuracy statistic, we sought to create an even better model via new neural networks and deep learning techniques; we worked to make a system to analyze movie posters via a convolutional neural network on multilabel classification. Per guidance of our TF, we first implemented a neural net from scratch. With knowledge on the limitations of multi-label classification and a dataset with class imbalance, we tuned and picked our parameters carefully. Our first model was comprised of 12 hidden layers and an input/output layer (4 2D conversion, 4 max pooling, 1 flattening, 3 dense and 2 dropout layers) and utilized a custom loss function based on weighted cross entropy. Then, using the Nesterov Adam optimizer, we rough tuned the model for 17 epochs (we stopped early when there was little accuracy improvement). We then fine-tuned our model for 9 epochs ending with a validation set accuracy of .573. Our second model was based on a previously created and trained neural network called the "Inception V3 pre-trained network". We added output layers to better address our specific task. This model utilized a categorical cross entropy and Geoff Hinton's RMSprop adaptive learning rate that, "divides the learning rate by an exponentially decaying average of squared gradients." Because this was a pre-trained model, we initially only tuning the output layers that we added. Then we tuned the top 2 inception blocks helping us to a final validation set accuracy score of .592. Below are visualizations showing its performance compared to our SGD and random forest models across genre (see Figure 2).

As you can see from the f-1 score accuracy visualization, the CNN and V3 deep learning neural networks consistently out performed both the Linear SVM with a SGD learning function and random forest model across most genres. It is worth noting that the SVM/SGD model did perform better in 4 genre categories and consistently outperformed the random forest model while still realizing fairly accurate results. If given more time, we would like to further develop our deep learning methodologies but also explore traditional machine learning techniques given the SGD model's performance. As we have learned throughout the semester, there is a tradeoff when using a neural network given the amount of data you have and time to train when compared to a machine learning technique. Our machine learning models were based on overview and crew/actor/director data whereas our neural networks were based entirely on poster image recognition. We were very happy with our results but when creating a final model, we would like to combine these models so they can help highlight and account for each other's weaknesses. Because they are trained on very different data sources, it provides an excellent opportunity to take advantage of the immense data resources available online.

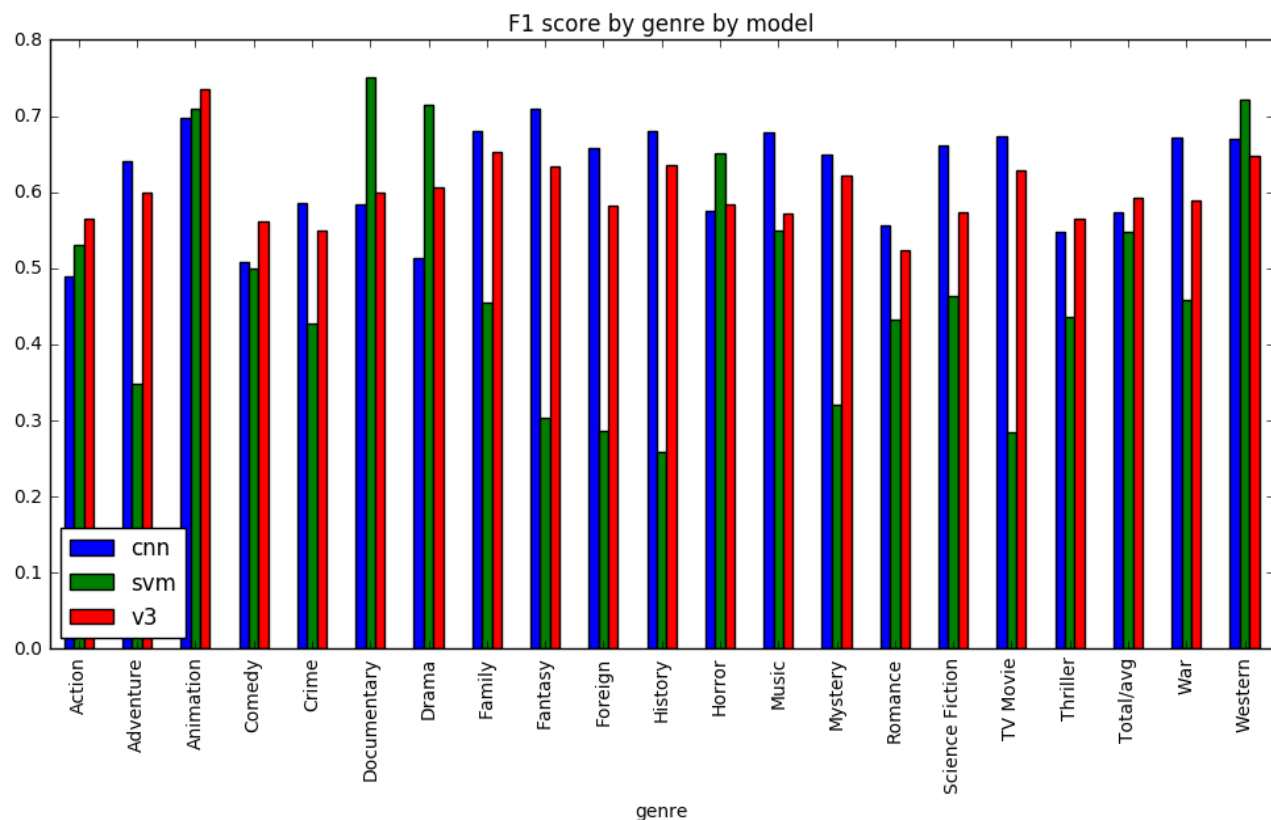


Figure 2. F1 score by genre by model.

Conclusion

We were able to achieve good results with both traditional methods (SGD/SVM classifier) using text information (plot/cast/crew) and neural networks (from scratch convolution network and pre-trained Inception v3 network) using poster images.

A slightly better result was shown by a pre-trained Inception v3 network (average f1-score was 0.592).

It looks like modern data science techniques like convolutional neural networks have an unparalleled ability to extract knowledge from massive data sources. In our experiment both pre-trained and custom CNNs performed better than the SVM model. It almost seems counter intuitive considering how much information the SVM model was given. Cast, crew, plot and overview should carry a lot of information about the genre a than posters. And yet the CNN models performed better. **CNNs are amazing.**