CS109B Final Project Report

Luke Heine, Stephen Turban, Noah Yonack, Danny Zhuang

Abstract

We set out to predict movie genres based on movie poster images and movie metadata. We trained many models, including Random Forest, LASSO logistic regression, Convolutional Neural Networks (built from scratch and pre-trained), and Recurrent Neural Networks. We used a dataset of 7,000 movies scraped from TMDb. After tuning all the models, we achieved our best performance with Random Forest using a bag of words model on the movie summary; this model had an accuracy of 93% compared to a baseline of 83%. By and large, the neural networks were surprisingly not able to beat baseline accuracy, which is the accuracy that a trivial classifier would achieve; we suspect that this is potentially because we did not have enough data relative to the sparsity of the genre label matrix.

Introduction

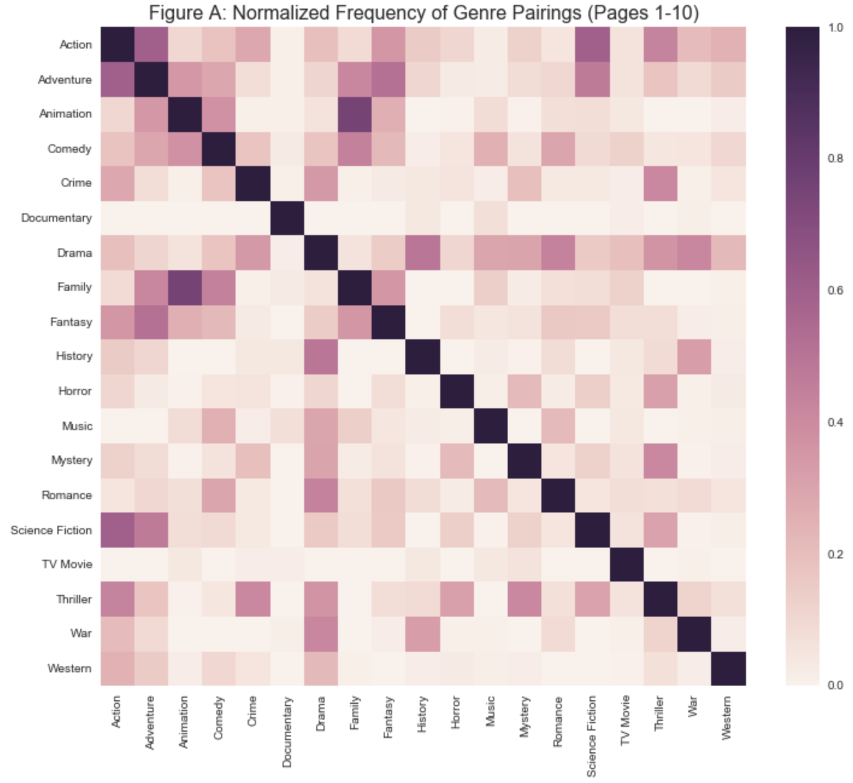
We are interested in using a movie's poster image and metadata to predict its genre labels. The results of this project have *substantial* industry application. Suppose that some movie genre combinations have posters with very distinctive artistic elements: for example, action-adventure movies, which may usually have posters with streaks of blood, or comedy-romance movies, which may usually show sunshine. With the ability to classify genres based on movie data, movie databases would be able to automatically generate genre suggestions that would be either rejected or approved by a final human check – saving the time of employees/contributors and thus enhancing workplace efficiency. This is similar to the modern news industry in which NLP algorithms suggest a few story headlines from which busy journalists can choose.

Data Collection

We chose to exclusively use the TMDb database instead of the IMDb database because of its easy-to-use API. Furthermore, TMDb is a crowd-sourced movie database that has an actively engaged community through its message board-functionality (IMDb recently discontinued its message board and is not crowd-sourced). We believe that an actively engaged online community that is emotionally invested in the quality of the movie data ensures a high level of data quality as online communities grow. Incidentally, this also helped us to solve the issue of TMDb and IMDb classifying the same movie with different sets of genres; though the two sources may store different sets of genres for an individual movie, we trust the quality of TMDb more.

We gathered our poster images and metadata from TMDb. For each genre, we pulled 20 pages of movies in order of descending popularity. The genre data came in text form (e.g. 'Action' or 'Adventure') – we converted this text into NxC count-vectorized matrices where N is the number of data points and C is the number of unique genres seen in the data. There are a total of 19 unique genres (C = 19). In **Figure A** below, we show genres that frequently occur together. We see there are genres such as Drama that occur very frequently with all the other genres, and there are genres such as Documentary that very rarely occur with other genres. We also notice from the lighter-colored squares that our response matrix of dummy-encoded genres will be very sparse.

Figure A: Normalized Frequency of Genre Pairings (Pages 1-10)



For our predictors, we download the movie poster images, text-based data such as movie descriptions, and metadata such as the release date. We also downloaded directors and lead actors, which was **not required by our rubric but we thought this data would add predictive value** as certain directors (e.g. Michael Bay) may make more films of a certain type (e.g. Action). Discussed in detail below, we use the poster images for our Convolutional Neural Networks (CNN) and use the text data in our Recurrent Neural Network (RNN) and traditional ML models (Random Forest & Lasso).

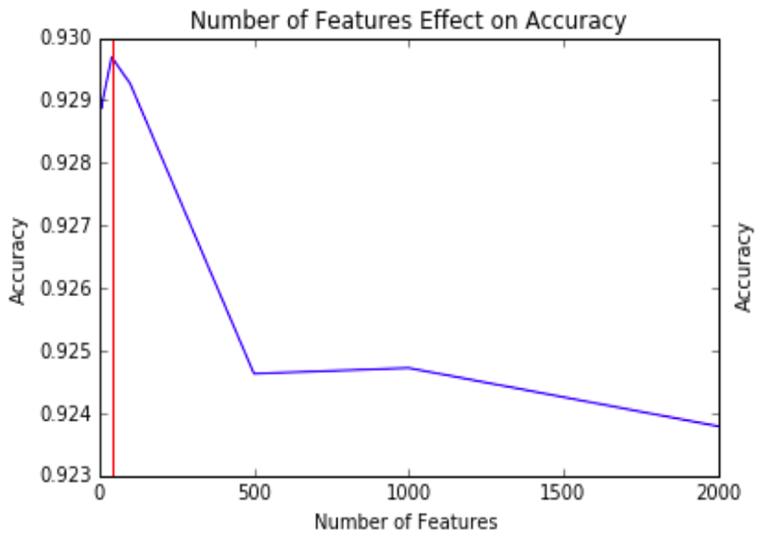
To train our initial Random Forest and LASSO models, we used stratified random sampling to preserve the genre frequencies of our original large dataset while still scaling down our data so that computation became tractable. However, once we began using AWS, we had the processing power to be able to train and test all of our models on the larger dataset, and there was no need to operate on a stratified sample.

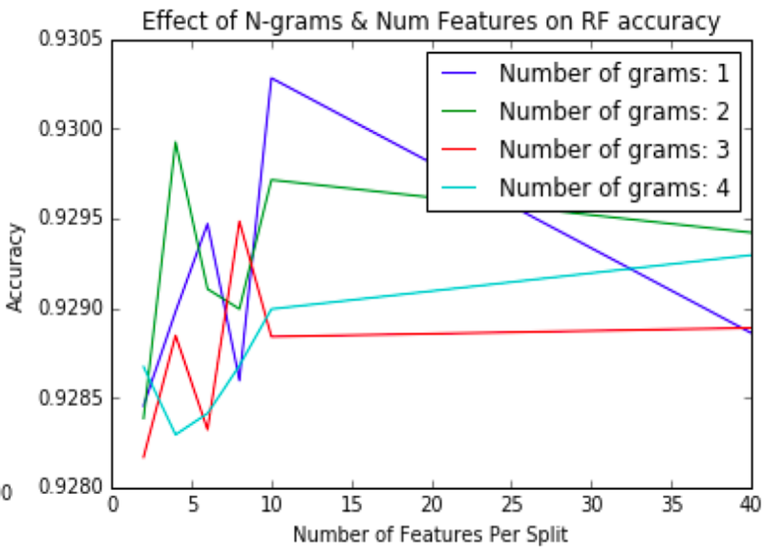
Modeling

Traditional Methods - Random Forest (RF) and LASSO:

We use a TF-IDF Vectorizer in order to convert the movie overview (essentially a short summary of the movie) into a matrix similar to that used in a plain bag-of-words framework (counts up occurrences of M unique words). However, the TF-IDF Vectorizer differs in that it assigns larger values to a word for a given entry when it occurs significantly more frequently in that entry that it does in the other overviews (and vice versa for smaller values). We also eliminated stop words such as “the,” “and,” or “but” because we believed these words would not provide predictive value. Intuitively, we believe that using the movie description will provide a large, useful feature set because there are many distinctive words - such as “gun” and “war”, that we would expect to occur in the action genre but not the family genre.

For our **RF model**, we tune the "max_features" parameter over a wide range of values from 2 to 500. We choose to use default values of "num_estimators=10" and "max_depth=None" based on our prior experience tuning random forests, where we found they do not significantly change model performance. We use a One-vs-Rest Classifier in order to perform multi-label classification with the RF. Our best results are when "max_features=10," with an accuracy rate of 93%. The accuracy over all tuning parameters is shown in the graph below. We also explore hamming loss, but choose accuracy given its high correlation to hamming loss and its simplicity in interpretation. We also explore using bi-grams, tri-grams, and quad-grams; however, we ultimately find that our more simple unigram model performs best.





For our **LASSO model**, we tune the regularization penalty over a range from 0.9 to 0.001. We choose to use the LASSO regression because it reduces dimensionality, which is a concern because our vectorized summaries have over 10,000 features for a dataset of 7,000 data points (p >> n). However, the LASSO just barely beats the baseline accuracy of predicting all zeros (85% > ~83%), regardless of whether or not we reduce the dimensionality of our data with Latent Semantic Analysis (LSA).

Convolutional Neural Networks (CNN):

CNNs traditionally perform better on image-based problems because they mimic human vision more closely than traditional machine learning methods.[1] Our features consist of movie poster data. We created multiple training sets, all with copies of the original movie poster which we then manipulated by size (32x32 pixels or 64x64 pixels) and color scale (RGB or greyscale). We built 5 different types of CNNs that varied in the number of convolution/pooling layers, filters, and kernel sizes.[2] We discuss results from our first and third models below because they were the best performing. Model 1 used 1 convolutional/pooling layer with 8 filters of size 3x3. Model 3 used 4 convolutional/pooling layers with 32 filters for each layer of size 2x2 per layer. We also

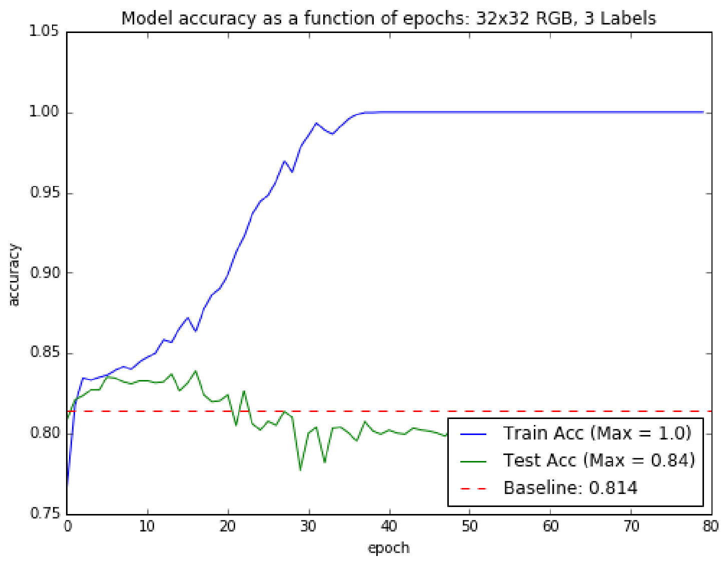
[1] http://cs231n.stanford.edu/

[2] A complete list of our models and their relative performances can be found in Milestone 4.

tried a handful of other models which varied in the number of layers, filters, kernel sizes, and activation functions but found that these two models were tied for best performing.

With 19 labels, none of our CNNs can meaningfully beat the baseline accuracy of ~83%. To make our multi-label prediction problem more tractable, we collapsed our 19 genres into 3 more general categories: *heartbeat*, which describes movies that tend to make your heart race (action, war, drama), *lighthearted*, which describes movies that tend to be fun and lighthearted (comedy, family), and *other*, which describes genres that aren't in *heartbeat* or *lighthearted* (western). In this new label space, we finally begin to see relative outperformance of 3 percentage points over our baseline accuracy. However, the maximum accuracy evaluated on a test set remained tightly contained between 83% and 84%, suggesting that there's not a lot of additional predictive ability gained from altering image size, color scale, or number of convolutional layers in the CNN. That said, it's possible that a significantly deeper CNN could beat our current performance.

The major factor that allowed our models to outperform the baseline was switching from 19 to 3 labels. This is likely because the original 19-label matrix is incredibly sparse (83% of the cells are zero), so we are effectively searching for needles in a haystack. We would likely need more training data to adequately find patterns in such a large label space.

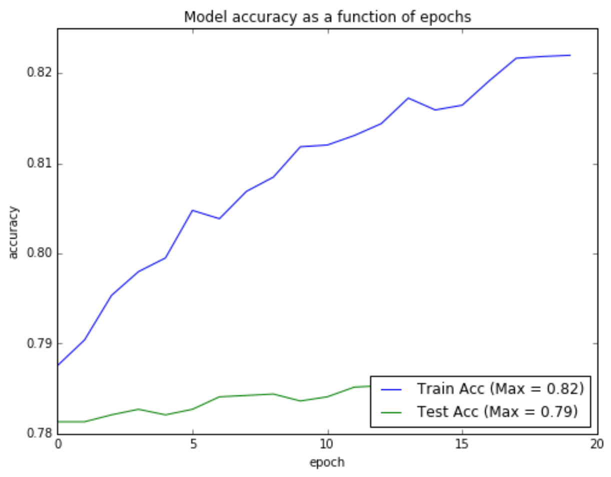


Model	**Image Size**	**Color**	**# Labels**	**Max Test Accuracy**	**Baseline Accuracy**
1	32x32	Greyscale	19	0.85	~0.83
1	32x32	Greyscale	3	0.83	0.814
1	64x64	Greyscale	3	0.83	0.814

1[3]	**32x32**	**RGB**	**3**	**0.84**	**0.814**
1	64x64	RGB	3	0.83	0.814
3	32x32	Greyscale	19	0.85	~0.83
3	32x32	Greyscale	3	0.83	0.814
3	64x64	Greyscale	3	0.83	0.814
3	32x32	RGB	3	0.84	0.814
3	64x64	RGB	3	0.84	0.814

Recurrent Neural Networks (RNN):

Although not required, we chose to explore RNNs on our own because they traditionally perform well on text and language based problems.[4] As a result, we use all our available text data columns: “director”, “lead actors”, “overview”, and “title”. We vectorize this text data into a matrix of counts over all the unique words. We train and test models on 19 labels as well as 3 labels as we did above for CNNs. When we trained on 3 labels, our RNN has the following accuracy as a function of epochs:



In contrast, our 19-label RNN has very little improvement as we iterate through epochs. This is potentially because, as above, our original 19-label matrix is too sparse relative to our data size to be able to model meaningful patterns in the data. However, our 3-label matrix is able to achieve 80% accuracy with an 81% baseline. We see from this that our model can make relatively informed estimates of the type of movie from the text associated with it.

Pretrained Neural Network (InceptionV3):

[3] Best Model

[4] http://karpathy.github.io/2015/05/21/rnn-effectiveness/

We also tried using the Keras pre-trained neural network InceptionV3 — a very deep neural network with 312 layers that took a very long time to re-train (~30 minutes of training time on AWS). Performance-wise, we were unable to push performance above ~85% even after varying the number of epochs, the learning rate, and the momentum.

Results

Our **Random Forest unigram model** using only our movie summary data "overview" does the best of any of our models. It predicts with roughly 93% accuracy which is 8 percentage points above our baseline score of 85% (from predicting all 0s on our 19-label matrix). Our models using neural networks (CNNs and RNNs) are unable to surpass 86% accuracy. This indicates that we are either not training with enough data on our neural networks or that random forests are better suited to this problem. Furthermore, this also indicates that a movie's summary contains a large amount of predictive ability on the movie's genre.

Conclusion

We were able to build an informative model (random forest) for predicting a movie's genre based on its summary. This is not only an intuitive result (an action movie might have more words related to combat), but also a useful one as it allows movie database platforms such as TMDb to recommend genres for a certain movie based on a movie's crowd-sourced summary. This makes the platform more interactive and thus more enjoyable for its user base.

Ultimately, we found that our neural nets were poorly performing relative to baseline accuracy. This is likely because our original 19-label matrix is too sparse relative to our data size to be able to model meaningful patterns in the data. Future work includes scraping *significantly* more data which we can then feed into our neural nets, which should theoretically perform well given their ability to, say, mimic human vision.