CS109b - Final Project Report

Team 14 - Dominic Murphy and David Modjeska
Harvard University, Spring 2017

Introduction

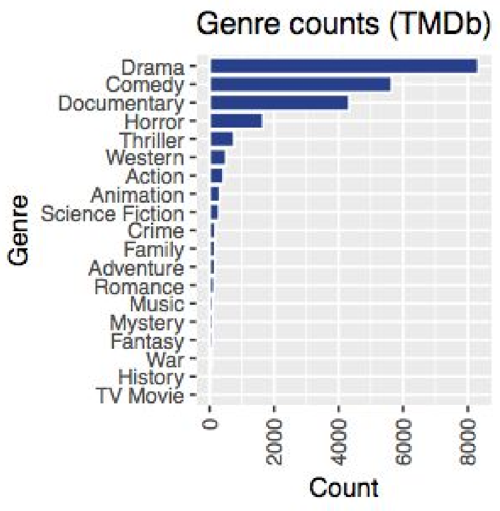
Movies are fascinating worldwide, and a movie's genre is one of the most interesting aspects. How do genres relate to movie plots, cast/crew, and posters? These are questions that a movie buff, industry reviewer, or data scientist might wonder about. With modern deep learning software and GPU hardware, opportunities exist to pair curiosity and technology for interesting, informal research.

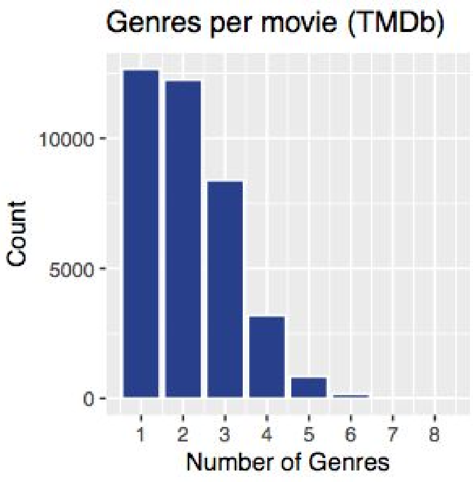
In the research domain, substantial effort has been devoted to deep learning techniques and parallel processing hardware. At the same time, these areas are largely new to the average computer-science student. In this report, we explore the genre question above using classic machine learning and modern deep learning techniques. Our deep learning was supported by Amazon Web Services (AWS).

Approach

The first phase of our project involved research and data acquisition. We investigated data sources, access, and features. On this basis, we chose a 40,000-movie research dataset from MovieLens. This set was used to acquire movie plot, cast/crew, and genre metadata from IMDb. Similarly, we acquired plot and genre metadata from The Movie Db (TMDb), as well as movie posters.

To get a sense of the data, we naturally performed exploratory data analysis (EDA). Key visualizations included the distribution of genres among movies, and the distribution of genre-tag counts among movies. Certain genres were much more common than others, for example, drama and comedy relative to thriller and western. The most common number of genre tags per movie was one or two.





Having obtained and gained a sense of the data, we modeled it using machine learning and deep learning methods. The goal was to classify genres from movie plots, cast/crew, and posters. For prediction with text and structured data, we implemented multiple methods, including including random forests, quadratic discriminant analysis (QDA), gradient boosting, and stacked ensembles. For prediction with image data, we implemented convolutional neural networks (CNNs).

With such a range of models, evaluation with multiple metrics was essential. We used the following metrics: accuracy, precision, recall, Hamming loss, percentage of positive predictions, and top-class success. (Top-class success is the percentage of movies where the genre with the highest predicted probability is among a movie's genre tags.) Accuracy is reported largely for completeness, since the sparse-matrix issue led to an apparent ceiling for accuracy.

As a preview of results below, we found that classic machine learning was effective for predicting genre from movie plots and people. Text mining performed best. Multi-label and multi-class approaches were both possible, but the multi-label approach was more natural. Minority genres were challenging for model performance. Fortunately, up-sampling and class-weights improved performance with these classes. Finally, deep learning presented logistical and technical challenges, but also power for inferring model definition from image data. Overall, our best metrics among various models included accuracy of 89%, precision of 74%, recall of 45%, and Hamming loss of 10%. We believe that a productive path for future exploration includes ensemble models, and multiple data types with one method.

Deep Learning

Our team built two Convolutional Neural Networks (CNNs). The first was trained from scratch, while the second was retrained from the VGG-16 network. For both, binary cross-entropy was used as the loss function for hidden layers, with sigmoid activation for the output layer, to support multi-label prediction.

The from-scratch network used a reduced version of the VGG-16 architecture, but without the benefit of pretraining. It was challenging to initialize weights of a new, deep model using only 40,000 movies. Despite using a glorot initializer, results were unsatisfactory. The network was too conservative, and it made no positive predictions for most classes. Our dataset was ultimately too small for the task, and our time constraints too tight.

Pre-trained CNN Metrics	
Accuracy	0.8826
Recall	0.2637
Precision	0.4110
Hamming Loss	0.1359

The second model was also based on the VGG-16 network. Significantly, though, this network used pre-trained weights. Only the top three layers were modified and re-trained. This approach provided beneficial information transfer. While 40,000 records was insufficient for a from-scratch network, they were sufficient to retrain the top three layers of the pre-trained network. With most layers frozen, this model trained three times faster per epoch, which allowed for more learning in smaller increments.

Because the pretrained model was more mature, a slower initial learning rate of .001 was selected. Initial momentum was set to 0.5, but the Nesterov method increase it gradually towards 0.99. A smaller image size of 128x128 (rather than the default of 244x244) was chosen to address memory, CPU, and GPU constraints. The final two dense layers where reduced from VGG's 4,096 channels to 1024 to reflect the smaller image size. In addition, the output layer was trimmed from 1000 features to 19 for movie genres. (Full details of our deep-learning hyperparameters are available in the project's Milestone 4 report.)

Classic Machine Learning

In implementing classic machine-learning, we pre-processed data differently for plot and people models, and then we used a single approach for model fitting and scoring. For the language model, we applied term-frequency/inverse-document-frequency (TF-IDF) vectorization to movie plots from IMDb. Unigrams offered better performance than did bigrams and trigrams. For the people model, IMDB's many-to-many relationships between cast/crew and movies were analyzed into a movie-person matrix. This matrix was analogous to a document-term matrix. Then models were fitted and scored with a single approach.

In order to reduce dimensionality for prediction, principal component analysis was applied to both models. 100 components worked well for most methods, though quadratic discriminant analysis performed best with only 10 components.

Several methods were used to fit models. For both movie plot and people models, the best methods were random forests, QDA, and gradient boosting. Other methods included decision trees, LDA, KNN, naive Bayes, and stacked ensembles. Some methods didn't make positive predictions for each movie: for these methods, we used the predicted probabilities to choose the most likely genre where needed.

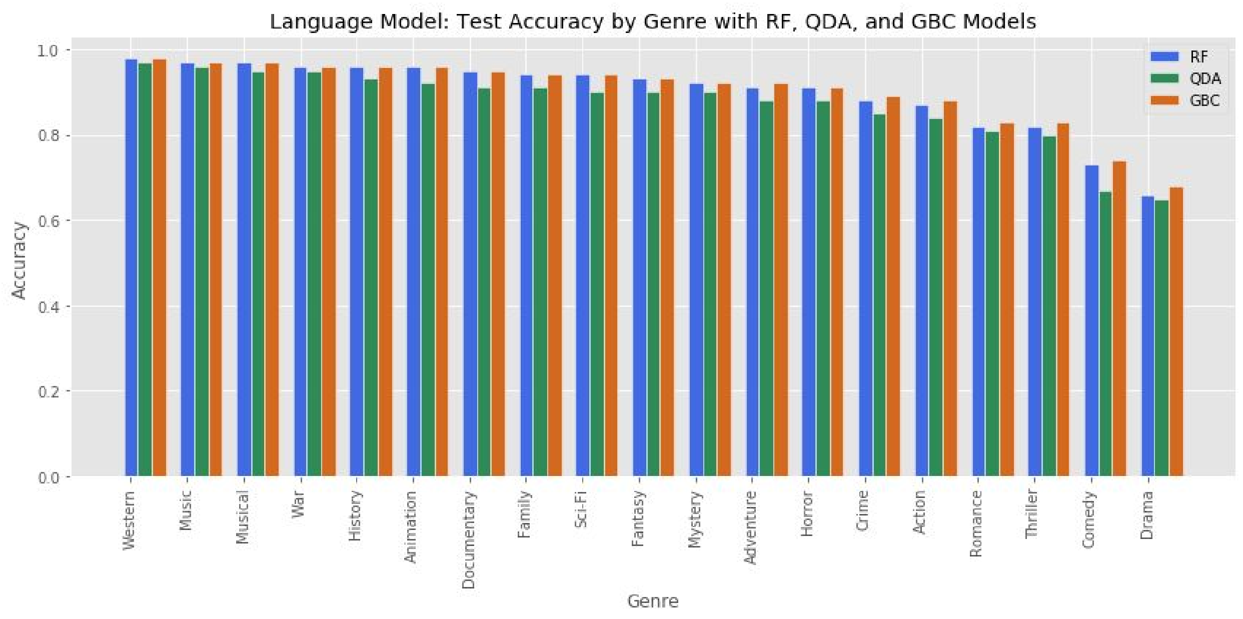
To evaluate model performance, we used a range of metrics for maximum understanding. These metrics included accuracy, precision, recall, and Hamming loss. Metrics generally identified boosting as the most effective method for the people model, but different metrics supported different methods for the language model (as shown in the Results section).

For further refinement, we would suggest tuning method hyperparameters with a grid, perhaps with one hyperparameter set per genre. To refine the people model, it may be possible to use a technique analogous to TF-IDF. Additional metrics (such as ROC curves) might also be useful.

Results

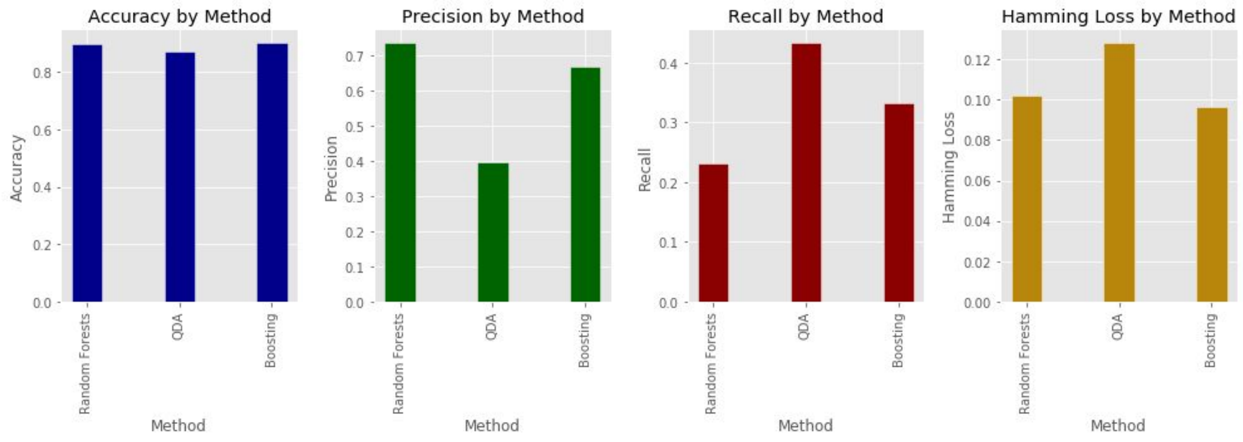
Overall, it appears that model performance was best for the language models, next best for the people models, and third best for the neural networks. Results are shown in the figures below.

Prediction accuracy varied widely across genres, with the most common genres having the lowest accuracy. Informally, a common genre tag is less informative than a rarer one. To demonstrate this variation, accuracy is plotted below for the three best language models.



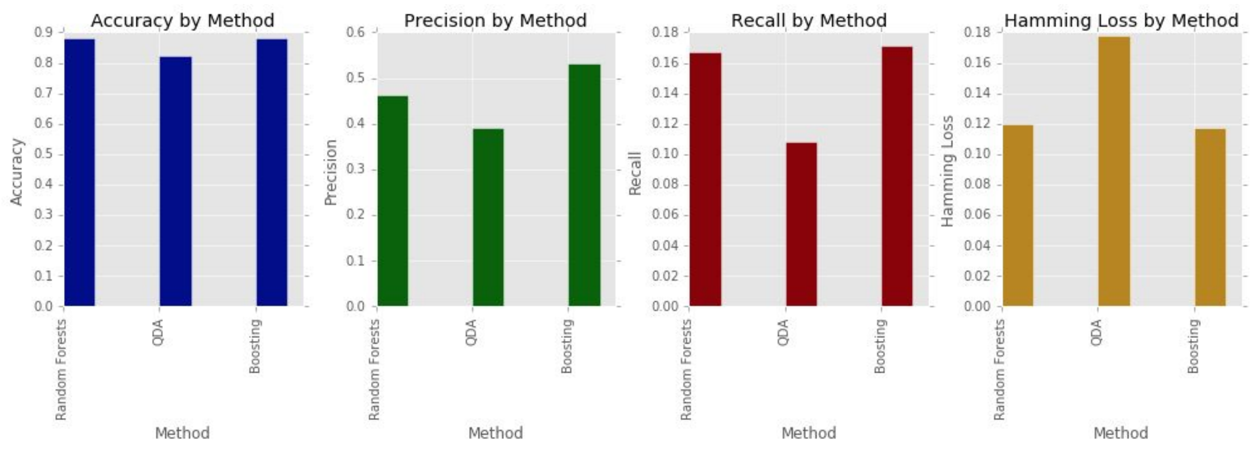
Among the best language methods, random forests and boosting showed the highest precision, while QDA showed the highest recall. The stacked ensemble (not reported here) seemed to be most heavily influenced by random forests, with similar results. Graphs below show metrics for the language and people models. While the language model generally performed better, the overall patterns are similar.

Language Model Metrics

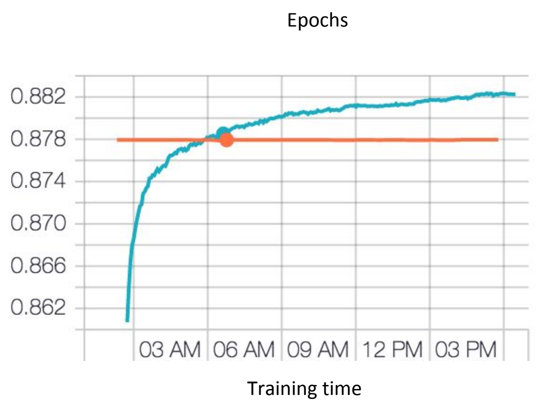


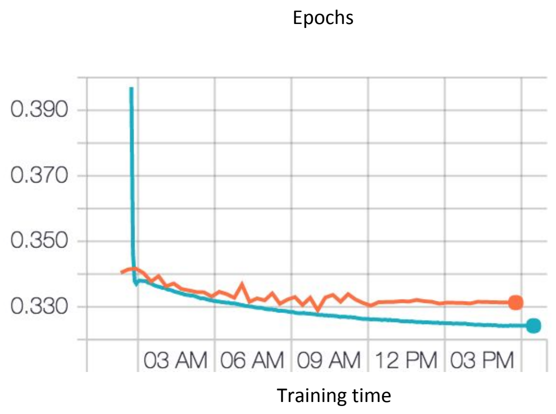
Among neural networks, the pre-trained model significantly out-performed the from-scratch model on precision, recall, and Hamming loss. The learning inherited from the larger dataset and longer training time of the pre-trained model presumably explains this difference. Figures below show validation accuracy and loss. These graphs were produced using TensorBoard with Keras training logs.

People Model Metrics



Despite sharing structure and many hyperparameters between the two deep-learning models, the from-scratch model proved challenging and slow: it often provided no positive predictions. For both models, validation accuracy seemed to approach null-model accuracy as a ceiling. The pretrained model improved over time (not reported here) and stabilized above null-model accuracy. Notably, the pre-trained model showed greater accuracy with minority classes. Nevertheless, validation losses were consistent between the models. During 24 hours, the pre-trained network trained to completion (presumably because of frozen layers), while the from-scratch model trained at half the speed.





Across classic machine learning and deep learning, most of our models showed an accuracy ceiling of 89%. This value matches the accuracy of a null model that always predicts zeros. The sparse matrix that contains the response variables leads to insufficient rewarding of positive prediction, in the absence of class weights or up-sampling. The best models climbed a few percentage points above this ceiling - these models included the pre-trained neural network, as well as the language model using a random forest.

Discussion

The combined toolkit of Keras, TensorFlow, AWS, and S3 offer tremendous power, range, and stability for deep learning. Still, a newcomer to deep learning should allow ample time for pragmatic learning and concept assimilation before tackling any real projects. For neural networks, learning rate and momentum were the most important hyperparameters, as expected. In addition, values for batch size, image size, and number of epochs impacted modeling results.

Tuning CNNs remains challenging, as research in this area is ongoing - parameter ranges are less well defined than with classical machine learning. Also, lengthy computations limit chances for fine tuning.

The multiplicity of available and relevant metrics gave conflicting feedback on the effectiveness of models. This situation required careful tradeoffs during project development. A key benefit was this: by employing more metrics, model performance could be tuned in a more informed way.

As mentioned earlier, the decision to adopt similar structures for the language and people models brought significant benefits, particularly acceleration and integration for project development.

Given more computational time and resources, we would explore a greater range of parameters, particularly those for dropout regularization. Similarly, we would explore ensembling opportunities more deeply, as well as ways to combine multiple data types in a single model. After all, movie plots, cast/crew, and posters all effective predictors. (During the project finalization stage, we tuned hyperparameters for the pre-trained neural network, and we implemented stacked ensembles with classic machine-learning methods.)

Conclusion

The approaches described here should be relatively portable across natural languages and application domains, with appropriate modifications. These approaches may suit a number of multi-label use cases.

As the challenge of classification grows with data volume, methods like the ones described in this report will gain in importance. Social tagging is useful for certain problems, and machine learning holds promise for scaling up. Social and automated methods may prove complementary in the long run.

Given the many available metrics for tuning model performance, prioritizing them depends on use case. In this report, the use case was abstract. So the project focused with relative neutrality on a range of metrics. In the real world, certain problems may demand high precisions (such as satisfying a consumer), while other problems may demand high recall (such as complete classification by a provider).

Classic machine is relatively mature, while deep learning is more recent to exit the pure research stage. As computing hardware and deep learning tools improve, one would expect that the logistical challenges of deep learning will be somewhat reduced. With luck, deep learning will eventually become as standard as classic machine learning in the toolkit of a data scientist.