
Deep Tomato

Leslie Tu, Petra Grutzik, Kate Park
Department of Computer Science
Stanford University
leslietu/pggrutzik/katepark@stanford.edu

Abstract

We predict Rotten Tomato movie critic ratings by extracting movie posters and text meta data from movie profiles and feeding these visual and text-based features into a deep neural network. In this paper, we scrape and parse Rotten Tomato movie profiles to build a dataset of 19 thousand movies and their information including poster images, descriptions, title, genre and Rotten Tomato critic rating. We compare several models including a text-feature based LSTM with GloVe vectors, AlexNet, custom Conv2D and combined LSTM-Conv2D model. We find that our combined model performs best at a test accuracy of 66.87 percent narrowly beating our LSTM at 65.61 percent test accuracy.

1 Introduction

Before sitting down to watch a movie, we often check its critic rating as an evaluation of the movie's greatness. If there is no critic rating, we can judge a movie by glancing at its poster and reading information about the film such as its title, genre, and description. With this information, we can predict that *Forest Gump* will be a great movie and *Fifty Shades Darker* will not be as compelling. Given thousands of movies produced annually, we ask whether we can build a model to determine movie critic rating based on movie profiles using neural networks.

In building our model we will use visual features, the colored movie poster, and textual information: title, description, and genre of a movie to predict whether a movie has over or below a 70% Fresh rating on RottenTomatoes.com. The Fresh rating is the percentage of critics who gave it a "thumbs-up" and the threshold of 70% ensures our classification is roughly 50-50 split across two bins.

2 Related Work

Several papers have tackled the challenge of learning meaningful information from movie posters or book covers. Kjartansson (2017), Kuprel (2016), and Libeks (2011) are a few papers that predict movie, book or album genre based on image analysis. Kjartansson judged book *genre* from a book cover and the title in text form using a deep CNN with 20,000 samples. Kjartansson achieved 80-90% accuracy in the combined image and text model, concluding that text features mattered more than the cover artwork, suggesting we will need to focus on textual features in addition to the poster.

While most examples in literature are classification tasks predicting genre, we propose predicting a rating. There is also a class of papers (Morovic (2011), Hsu (2014)) predicting user ratings for movies based on content or collaborative methods by using ratings of similar users. However, this is less relevant for our task at hand which attempts to predict a collective score across all users rather than make a movie recommendation for a particular user.

Sun (2016) approaches a similar problem of predicting movie rating, fitting a Random Forest regression on 5 thousand movies from IMDB and finds that the number of faces in a movie poster has a non-neglectable effect to the movie rating. Sun’s final model has a mean squared residual of 0.89023. While fitting a regression would be more relevant, we approach the problem as a classification task (see Section 5) with a larger dataset of 19 thousand movies from Rotten Tomatoes.

Alex Krizhevsky (2012) classified images using a new convolutional neural net structure now known as the AlexNet. Krizhevsky et al. achieved top-1 and top-5 error rates of 37.5% and 17.0% which was quite better than previous state-of-the-art image classification models. This image classification network could be a promising framework for detecting specific features that differentiate images from each other. This ability could be useful for detecting features that differentiate posters of "Fresh" movies from posters of "Rotten" ones.

3 Dataset and Features

Our dataset comes from the Rotten Tomatoes website (<http://www.rottentomatoes.com>), which has about 19 thousand usable movies (21 thousand movies did not have Fresh rating). Figure 1 shows an example of a movie input.


title	poster	description	genre	percent fresh
Wonder Woman		An Amazon princess (Gal Gadot) finds her idyllic life on an island occupied only by female warriors interrupted when a pilot (Chris Pine) crash-lands nearby. After rescuing him...	Action, Adventure, Drama, Science Fiction, Fantasy	92

Figure 1: A sample movie input.

We used an 80/10/10% split to get 15,648 training examples, 1,957 dev examples, and 1,957 test examples.

Title and genre were our text features (we experimented with description, but it hurt our performance), and we used pre-trained GloVe vectors to encode the words.

For the images, we resized them to 224×224 , randomly brightened and saturated them, and clipped them within the usual range. We kept the colors, so we had 3 channels. After our model started over-fitting the train set, we introduced data augmentation by horizontally flipping the images.

4 Methods

We approached our problem of classifying movies with text models, image models and finally a combined model.

Initially, we approached the problem as a regression using mean squared error as our loss, however this was difficult to train (our dev mean squared error was 887 after training on 4K movies or about 30% off an average) and thus we moved towards a classification task. We attempted using the suggested ten equally sized bins of scores and softmax cross entropy loss. However cross entropy loss did not accurately capture the problem since the bins were not independent (classifying a movie into one bin when it was supposed to fall into the adjacent is not horrible). This fact combined with low performance on 10 bins motivated us to simplify the problem into a 2 bin classification task which we will use for the rest of the paper.

4.1 Text Model

Our first attempt at processing the text features used a 4 layer neural network, representing words as n-gram counts. We quickly abandoned this model for more standard natural language processing techniques. We represented words using pre-trained GloVe vector representations. We fed these word embeddings into a Long Short-Term Memory (LSTM) network to preserve word ordering of movie titles and descriptions. We zero-pad our word sequences to the length of the longest input, propagate the embedding layer through 3 LSTM layers each with 128 hidden units and conclude with softmax activation.

Our loss was cross-entropy, given by $-(y \log(p) + (1 - y) \log(1 - p))$, where p is the predicted probability and y is the true one-hot class vector.

Due to overfitting, we introduced dropout with keep probability of 0.4 after the first LSTM layer and 0.3 for the next two layers. We trained with the Adam optimizer with a minibatch size of 512 and learning rate of 0.001.

4.2 Image Model

We tried two models to learn on the posters. The first was AlexNet, and the second was a slightly smaller convolutional network. AlexNet did not work well for our problem, so we continued customizing our the second model.

Our final model starts with a 3×3 convolution with 16 channels, a stride of 1, and valid padding, followed by a max pool with stride 2 and padding 2. We repeat this block three more times, with the number of channels doubling each time. Then, we flatten the tensors to 25088-dimensional vectors and run them through two fully connected layers. Finally, we use softmax to get a 2-dimensional output where the entries indicate the probability of the class.

Our loss function was also cross-entropy.

We ran into over-fitting problems, so we added dropout with a keep probability of 0.8 in the fully connected layers. We trained with the Adam optimizer with a minibatch size of 512 and a learning rate of 0.001.

4.3 Combined Model

After training the text and image models, we linearly combined the softmax output layer. We froze the weights in each model, extracted the 2×1 softmax outputs of each, and used a weighted average $\alpha v_{text} + (1 - \alpha) v_{image}$ to produce the final prediction. We tuned the optimal weight α on the train set. Our final α was 0.6.

Figure 2 shows our combined model architecture.

5 Results and Discussion of Various Models

Figure 3 shows our results for various models. Accuracy is the percentage of predictions that were correct. We used this metric since our classification problem purposefully divides the dataset into two equal sized bins: above 70 or below 70 fresh rating. Figure 4 shows how our metrics for text and images changed over training epochs.

5.1 Text Model

The final GloVe vector LSTM on movie titles and movie genres performed surprisingly well with a train accuracy of 69.37% and test accuracy of 66.51%. To achieve this performance, we pulled out movie descriptions from our inputs because the feature drastically slowed down training and hurt performance. Using both movie titles and movie genres works best (movie genre without title leads to 64.8% accuracy on train).

Due to overfitting on the train set as shown in 4, we used early stopping at 17 epochs when the dev loss was lowest.

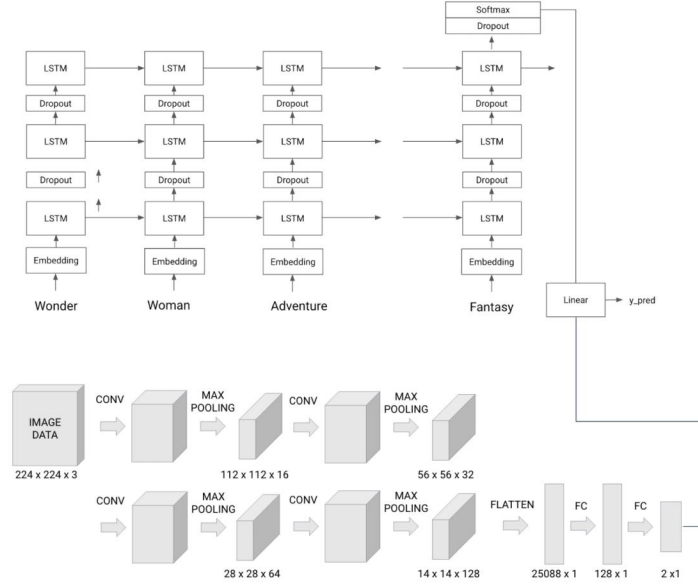


Figure 2: Architecture of combined text and image models.

Model	Train Acc. %	Dev Acc. %	Test Acc. %
Baseline (Random)	50	50	50
GloVe LSTM	69.37	67.59	66.51
Conv2D	61.22	56.59	56.85
AlexNet	53.9	52	—
Combined GloVe LSTM + Conv2D	69.80	67.99	66.87

Figure 3: Accuracy results of various models.

5.2 Image Model

Using AlexNet we obtained a 52.9 % accuracy on our train set and only a 52 % accuracy on our development set. This model was not performing well for our task, so we abandoned further testing. Achieving an accuracy near random baseline suggests the AlexNet is not a useful model for predicting the quality of a movie from the movie poster. Perhaps image classification, which AlexNet was built for, is too different from our problem.

The convolutional neural network with four convolutional layers worked much better than the AlexNet achieving a 61.22 % accuracy on the train set, a 56.59 % accuracy on the dev set, and a 56.85 % accuracy on the test set. The model was performing much better on the train set (upwards of 90% accuracy) than on the development set, so we used early stopping and saved the best weights as shown in the table, not the final weights.¹

5.3 Combined Model

After training our separate models and freezing their weights, we did a search over possible weighted averages and chose the weight that gave the best dev accuracy. We weighted the text output 60 % and the image output 40 %.² We found it interesting that the best weighting was almost even, given that the image model performed significantly worse than the text model.

¹Andrew Ng was impressed our tuned CNN performed better than the random baseline for this difficult task.

²In response to our project, Andrej Karparthy said that he ran into a similar problem processing Youtube videos in that the textual metadata provided most of the accuracy and the video/image data bumped their metrics by less than 1 percent.

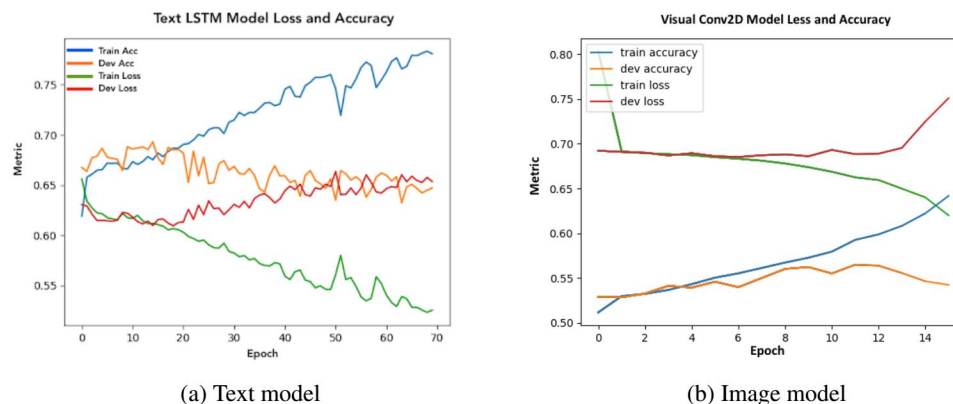


Figure 4: Model results over epochs

6 Conclusion

We found judging movie quality to be a difficult problem because of its subjective nature. When we challenged ourselves as humans to classify movies as having a Freshness rating greater than 70 or less than or equal to 70 based only on the movie poster, title and genre we achieved around 80% accuracy suggesting that the avoidable bias is smaller than we thought. Note that the Freshness critic rating is formed after watching the movie. While an LSTM network analyzing the textual information of the movies provided most of the most accuracy of the combined models (66.51%), the convolutional neural net trained on movie posters did increase the accuracy of the final combined model (66.87%) suggesting the movie poster does provide information about movie quality.

7 Future Work

In the future we would like to train a model that includes description in the text model. The words used in the description could strongly reflect the quality of the movie.

Secondly, we would like to further reduce the overfitting on the convolutional neural net. Next steps include a more detailed search of hyperparameters of the model and more recent complex CNNs that have been successfully used in image recognition problems.

Next, we would also like to do further research on combining the text neural net and image convolutional neural net models. Rather than combining the final softmax layers, we would feed in the output of the last layers preserving the vectorization of each input to the two neural nets into a third neural net with multiple fully connected layers. This may be able to combine the information from the text and image neural nets more successfully.

8 Contributions

We did the vast majority of our work sitting together in the same room, so everyone spent roughly the same amount of time on the project. Kate built the text models and ran regression experiments, Leslie scraped the Rotten Tomato data and implemented the AlexNet (and abandoned it quickly), and Leslie and Petra built and tuned the image model and the final combined model.

9 Code Repository

<https://github.com/lesliettu/cs230movies>

References

- [0] Krizhevsky, A. & Sutskever, I. & Hinton, G. E. (2012) ImageNet Classification with Deep Convolutional Neural Networks. NIPS: Neural Information Processing Systems, Lake Tahoe, Nevada. 2012.
- [1] S. Kjartansson and A. Ashavsky, "Can you Judge a Book by its Cover?," Stanford CS231N. <http://cs231n.stanford.edu/reports/2017/pdfs/814.pdf>. 2017.
- [2] B. Kuprel, "Judging a movie by its poster using deep learning." Stanford CS221. 2016.
- [3] J. Libeks and D. Turnbull, "You can Judge an Artist by an Album Cover: Using Images for Music Annotation," IEEE Multimedia 18(4):30-37. May 2011.
- [4] M. Marovic et al, "Automatic movie ratings prediction using machine learning," University of Zagreb. http://www.csc.kth.se/~miksa/papers/AutomaticMovieRatingsPrediction_MIPR0.pdf. 2011.
- [5] P. Hsu et al, "Predicting Movies User Ratings with Imdb Attributes," International Conference on Rough Sets and Knowledge Technology.pp 444-452. RSKT 2014.
- [6] C. Sun, "Predict Movie Rating," NYC Data Science. August 2017.
- [7] J. Pennington, R. Socher, and C D. Manning. "GloVe: Global Vectors for Word Representation." 2014.
- [8] M. Abadi et al, "TensorFlow: Large-scale machine learning on heterogeneous systems," Software available from tensorflow.org. 2015.
- [9] F. Chollet et al, "Keras," <https://github.com/keras-team/keras>. 2015.