Creating Aggregate Ratings of Amazon Movie Reviews using Deep Learning

Erika Deckter
edeckter@stevens.edu
Stevens Institute of Technology
Hoboken, NJ

ABSTRACT

This paper describes a novel method for creating aggregated movie review ratings using Natural Language Processing techniques. Aggregate movie reviews provide a single metric of the overall quality of a movie without having to review multiple potentially divergent reviews. Three models were built using linear regression, a recurrent neural network (RNN) and a one-dimensional convolutional neural network (CNN) to predict review scores on a continuous scale from 0 to 1 for individual reviews. The individual reviews for each movie were then averaged to create the aggregated review score. The models were trained using a subset of data from the Stanford Network Analysis Project (SNAP) consisting of Amazon user reviews of movies. As measured by mean squared error, the RNN model had the best performance for predicting both individual and aggregated review scores. The CNN model also performed well for a small number of epochs but quickly overfit. The linear regression model did not generalize well. The three models were then used to predict individual and aggregate review scores for out-of-sample reviews of recent movies.

KEYWORDS

movie ratings, natural language processing, neural networks

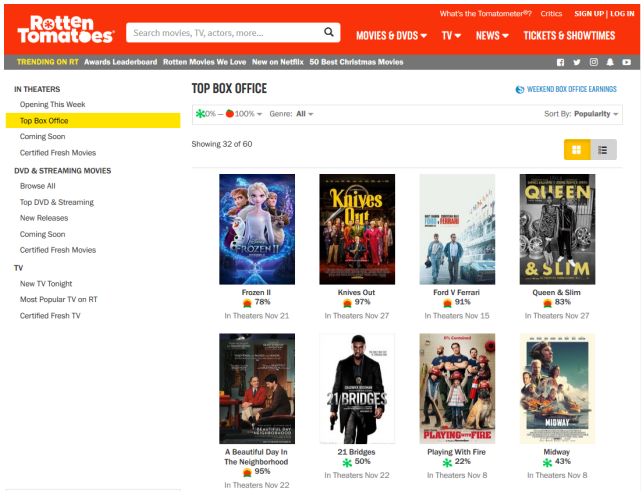

Figure 1: Screenshot of Rotten Tomatoes Ratings for Top Box Office as of December 7, 2019 https://www.rottentomatoes.com/browse/in-theaters/

1 INTRODUCTION

Because critics and other reviewers often disagree on the quality of a movie, aggregate movie reviews give busy viewers a quick overview of the overall quality of a movie without having to read through many reviews. Websites like Rotten Tomatoes provide aggregated movie review scores by assigning a binary sentiment label to each review for a film ("fresh" for positive and "rotten" for negative in Rotten Tomatoes' case) and taking an average of the binary scores to come up with an aggregated score between 0% and 100%.

For example, if a movie's reviews were all mildly positive, Rotten Tomatoes would assign a sentiment of positive to every review, resulting in a score of 100%. In contrast, if a movie had very polarizing reviews, for example, 50 very positive and 50 very negative reviews, this methodology would result in a potentially misleading 50% aggregated score.

Therefore, in this paper I am proposing a new method to aggregate movie reviews by assigning a score between continuous 0 and 1.0 to each movie review and taking the average of the review scores to create the aggregate movie score. Unfortunately, many publications do not append handy scores to their reviews in order to use them in aggregation. Human taggers can read each review and assign a score based on the content of the review; however, this is extremely inefficient when hundreds of movies are released each year and many publications write reviews of these movies. This is why I am proposing the use of Natural Language Processing (NLP) techniques to automatically assign a review score based on the text content of the headline and review.

Several models were trained to predict review scores using data from the Stanford Network Analysis Project (SNAP), which consists of approximately 8 million Amazon user reviews of over 250,000 movies [11]. These models included a linear regression model, a recurrent neural network (RNN) and a convolutional neural network (CNN).

2 BACKGROUND

Much of the existing work in the area of classifying movie reviews is focused on binary sentiment classification (see Bakliwal et al. [6] for an example). Socher et al. [7] did include fine-grained sentiment analysis using five merged classes based on the Stanford Sentiment Treebank dataset, although this is still a classification model.

Early approaches to sentiment classification for movie reviews used single words or n-grams tagged as positive or negative and simple counts of the frequency of these words to determine sentiment using common machine learning algorithms such as Naive Bayes and Support Vector Machines [12]. A major drawback to this single-word approach is that it cannot capture nuance in language

such as negation (for example, "not bad" has a very different sentiment that "bad" taken by itself). N-grams can partially alleviate the negation issue but still broadly ignore word order. In addition, n-grams cannot capture long-range dependencies in sequences of text. Neural approaches introduced the concept of dense word embeddings [8] as input features to traditional machine learning algorithms or neural networks.

Review aggregation sites such as Rotten Tomatoes generally use binary classification to calculate the single aggregated score for a movie. Reviews are classified by human "curators" whose job is to read movie reviews and classify them as positive or negative and to choose a representative quote from the each review [5].

My approach is to automate this task by first assigning a review score to each individual review from a continuous scale of 0 to 1 (rather than a binary or multi-category classification) and then creating a single aggregated review score for each movie by taking an average of the individual review scores for that movie. My hope is that this novel approach will provide more fine-grained indications of a movie's quality as compared to the aggregated score based on binary sentiment classification.

3 METHODOLOGY

There are four key stages in order to go from text reviews to aggregated review scores for each movie. The first stage is to preprocess the data in an attempt to eliminate user bias in ratings and to rescale ratings to be on a scale from 0 to 1. The next stage is to encode the text into vectors (e.g., word embeddings) so that the text can be used as input into a machine learning algorithm. The third stage is to predict the score for each review using a regression algorithm. The final stage is to take the predicted scores for each individual review and aggregate them into a single combined score for each movie.

3.1 Data Preprocessing

The SNAP dataset has ratings for each review on a scale of 0 to 5 stars, in 0.5 star increments. In order to eliminate the bias of a particular user, the mean and standard deviations of the reviews for each user were calculated and users' reviews were standardized as in equation 1:

$$\tilde{s}_{i,j} = \frac{s_{i,j} - \mu_j}{\sigma_j} \tag{1}$$

where i represents the movie index, j represents the user index, $s_{i,j}$ is the original user score and $\tilde{s}_{i,j}$ is the standardized score.

In order to get the standardized user scores onto the 0 to 1 continuous scale, min-max normalization was then applied to all standardized scores as in equation 2:

$$\bar{s}_{i,j} = \frac{\tilde{s}_{i,j} - \min}{\max - \min} \tag{2}$$

For users where the standard deviation is 0 (i.e., the user rated all movies in the training dataset with the same score), the standard deviation was defaulted to 1. This is both to avoid a divide by zero error and to convert the normalized score to simply be the difference between the user's mean score and the actual score for each review in these cases.

Figure 2 shows the range of normalized scores for the training and the test datasets. By definition the training dataset scores are between 0 and 1. There are some outliers that are lower than the training set minimum and higher than the training set maximum that make the normalized test set range go from approximately -0.75 to 1.5. In addition, the means of both the training and test datasets are 0.705. This means that more reviews in the dataset skew positive than negative. This may be related to the subset of data used for this project or it may be that users tend to take the time to review products they like more often than products they don't like.

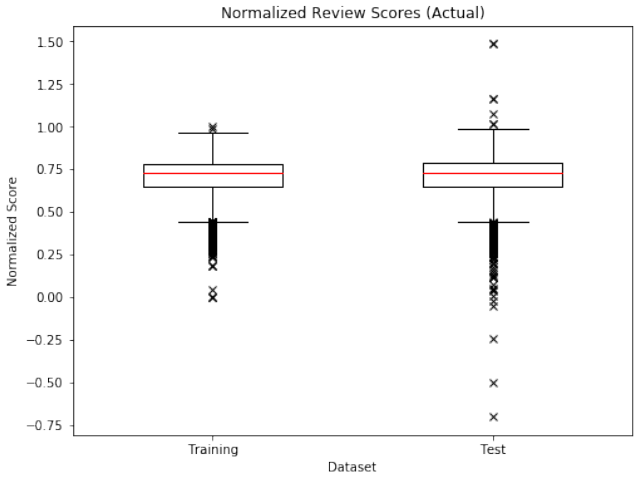

Figure 2: Ranges of Normalized Review Scores for Training and Test Datasets

3.2 Encoding Text as Word Vectors

In order to assign scores to each text review, first the text must be converted into vectors that the an algorithm can understand. One way to do this is to create a vector the size of the vocabulary, V for every word in the text. Words can be represented as one-hot encoding (i.e., a 1 in the corresponding row if the word exists in the document or a 0 otherwise) or term frequency where the number of times a particular word in the vocabulary appears in a document is recorded [10]. This gives the same weight to very frequent words as to rare words, but since rare words obviously give more information about a document than frequent words, it is better to weight the term frequencies by the inverse of the document frequencies for each term [9]. This is referred to as Term Frequency-Inverse Document Frequency or TF-IDF.

The methods described above lead to long, sparse vectors. An alternative is to use shorter, dense vectors called word embeddings. These embeddings often work better for machine learning applications. Word embeddings can be learned as part of a neural network (the embeddings for each word in the vocabulary are learned like any other weights matrix in a neural network algorithm) or pre-trained on a large corpus of documents such as in Word2Vec [8] or GloVe [13].

The review summary (headline) and the text of the review were concatenated into a single text field. This summary/text field was

then encoded using both TF-IDF and word embeddings. The embeddings were learned during model training (no pre-trained embeddings were used).

3.3 Assigning Review Scores

Three models were built in order to compare methods for predicting the review score.

3.3.1 Baseline Model: Linear Regression. As a baseline, a simple linear regression model was trained to predict review scores. The input to this model was the TF-IDF word vectors for each review and the output was a predicted review score. The advantage of this model is its simplicity: It is easy to explain both the methodology for converting words to vectors and the linear equation to predict review scores to a non-technical audience. The disadvantages are that a linear model may not be a good representation of the mapping from word vectors to review scores. In addition, using TF-IDF produces sparse word vectors which are inefficient from a memory standpoint.

3.3.2 Recurrent Neural Network. The second model was a bidirectional Long Short Term Memory (LSTM) recurrent neural network (RNN). The LSTM model is an improvement over the simple RNN in that it allows for more information from earlier in the sequence to be remembered in the output. Using a bidirectional model actually builds two LSTM networks: one that reads the sequence from the beginning to the end and a second one that reads the sequence from the end to the beginning. The output of the forward and backward LSTM layers are concatenated to produce a final output. This prevents the model from "forgetting" information seen early in the sequence. The advantages of an RNN model is that it can process arbitrarily long sequences while sharing weights across each time step and that it can represent the order of words in a sequence (as opposed to a "bag of words" model which does not have any concept of word order). The disadvantages are that without tricks such as bidirectionality and gated architectures, it is hard to represent the entire sequence in a single output. In addition, training of an RNN model can be slow.

The input to the bidirectional LSTM model is a set of dense word embeddings learned as part of the mdoel training. The output of the LSTM model is then fed to a dense layer with a rectified linear unit (ReLU) activation function to predict the model score. This is a regression problem so an activation function on the output layer is not strictly necessary; however, the ReLU activation function ensures that the final predicted score will be positive (since all negative values are set to 0 in the ReLU function).

3.3.3 Convolutional Neural Network. The last model was a one-dimensional convolutional neural network (CNN). A convolutional neural network applies fixed-length filters to a sequence of words in order to pick out meaningful patterns. The advantage of a CNN is that it can find key subsequences of words regardless of where the subsequence occurs in the text (via the sliding filter). The disadvantage is that it does not account for word order in a sequence beyond the length of the filter.

The input to the CNN model is a set of dense word embeddings learned as part of the mdoel training. As with the RNN model, the output of the CNN model is then fed to a dense layer with a rectified linear unit (ReLU) activation function to predict the model score.

3.3.4 Model Evaluation. All three models were evaluated using Mean Squared Error (MSE).

3.4 Aggregating Review Scores

The final stage of the process was to take the predicted scores of each individual review and aggregate them into a single score for each movie. This was done by taking a simple average of all the scores for a given movie.

4 EXPERIMENTAL DESIGN

The stages described in the Methodology section were implemented in Python using a Jupyter notebook. The sci-kit learn package was used to implement TF-IDF encoding as well as the linear regression model. Keras was used with a TensorFlow backend to tokenize and apply zero padding to the review texts as well as to implement the RNN and CNN models.

4.1 Data Filtering

While the SNAP dataset has over 8 million reviews, in order to train models in a reasonable amount of time on the hardware available, the dataset was filtered down to a small subset of approximately 215,000 reviews. The dataset was then further filtered to remove reviews from users with less than 10 total reviews and reviews for products with less than 10 total reviews.

The final dataset used for modeling had 37,642 reviews for 1,095 movies. The movies were then divided into training and test datasets using an 80% split for training and a 20% split for test. This gave 876 movies in the training dataset with 30,401 corresponding reviews. The test set had the remaining 219 movies with 7,241 corresponding reviews.

4.2 Text Preparation

The shortest review text was 5 words, while the longest text was over 4,400 words. The average review length was 308 words. Figure 3 below shows the histogram for review length.

The full concatenated summary and review text was used to create the TF-IDF matrices for each review. For the word embeddings learned as part of the RNN and CNN modeling, a fixed vocabulary of the 10,000 most common words was used. A special unknown (<unk>) token was also included for all out of vocabulary words. A fixed sequence length of 300 words was used as input to the RNN and CNN models. For sequences shorter than 300 words, zero padding was applied to the end of the sequence. For longer sequences, only the first 300 words were used.

4.3 Hyperparameter Tuning

The RNN model was training using a batch size of 32 with the RMSProp algorithm using a learning rate of 0.001 for 20 epochs. A random 20% of the training dataset was used as a validation set for hyperparameter tuning. Output sizes of 8, 16 and 32 were tested for both the size of the embedding layer and the size of the LSTM layer. The LSTM layer also included dropout and recurrent dropout values of 0.5 to help avoid overfitting.

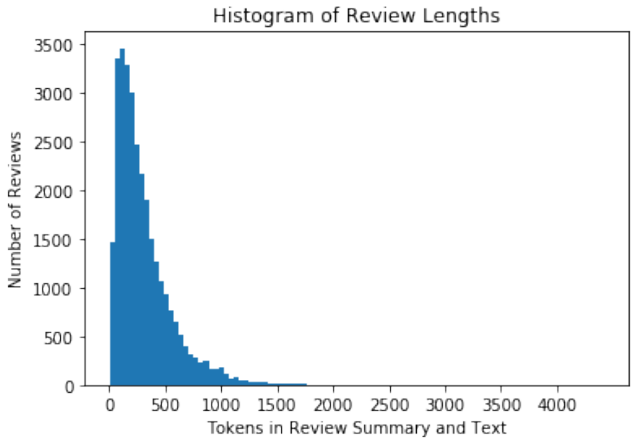

Figure 3: Histogram of Word Lengths for Concatenated Review Summary and Text for the Training Dataset

Figure 4 shows the training and validation losses for select combinations of embedding and LSTM layer sizes.

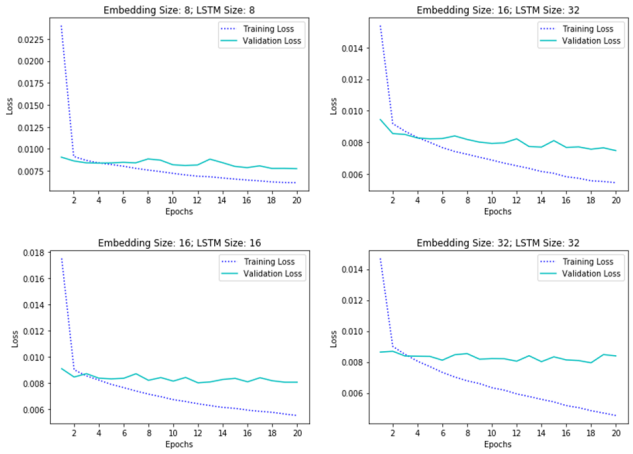

Figure 4: Training and Validation Loss for Select Hyperparameter Combinations of the RNN Model

Figure 5 shows the final architecture selected for the RNN model, consisting of an embedding layer of 16 output dimensions and a bidirectional LSTM layer of 32 output dimensions for the forward layer and 32 output dimensions for the backward layer, resulting in a final concatenated output of 64 dimensions. This model was trained on the full training dataset for 20 epochs.

The one-dimensional CNN model was training using a batch size of 32 with the RMSProp algorithm using a learning rate of 0.001 for 20 epochs. A random 20% of the training dataset was used as a validation set for hyperparameter tuning. Embedding output sizes of 8, 16 and 32 were tested, in combination with filter counts of 5, 10 and 15 and filter sizes of 5, 7, 9 and 11. Filters used a stride of 1 with zero padding set to "same."

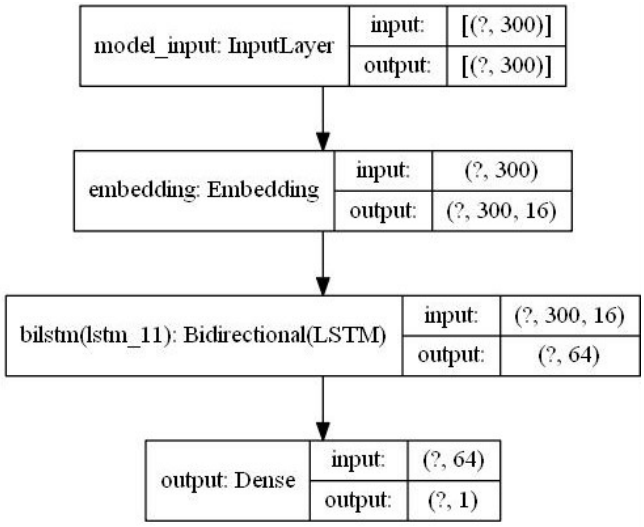

Figure 5: Recurrent Neural Network Architecture

Figure 6 shows the training and validation losses for select combinations of embedding layer sizes, filter counts and filter sizes. Figure 7 shows the final architecture selected for the CNN model,

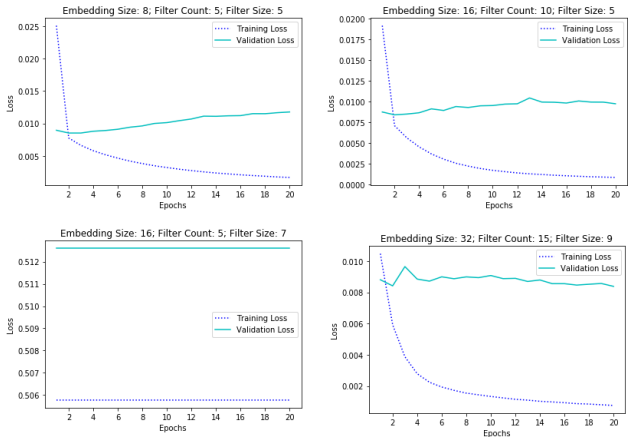

Figure 6: Training and Validation Loss for Select Hyperparameter Combinations of the CNN Model

consisting of an embedding layer of 16 output dimensions, 10 filters and a filter size of 5. The CNN model quickly overfit, so the final model was trained on the full training dataset for only 3 epochs.

5 EXPERIMENTAL RESULTS

Table 1 below shows the mean squared error (MSE) of the training and test datasets for the three models.

While the baseline linear regression model has the best training MSE, it has the highest test MSE. This is an indication of overfitting, as the model does not generalize well to unseen data. Of the two neural network models, the RNN has the best training and test MSE. The RNN also seems to be a more stable model, with a relatively

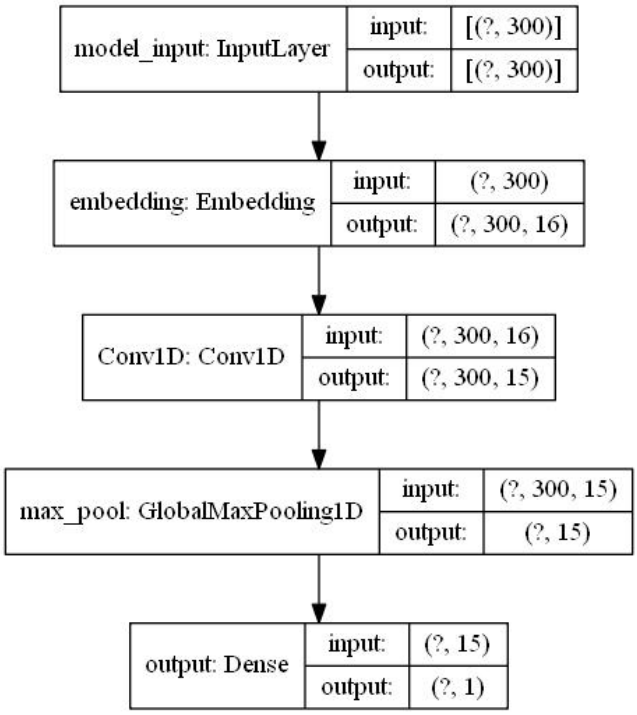

Figure 7: Convolutional Neural Network Architecture

Table 1: Individual Review Score Mean Squared Error

Model	Training Dataset	Test Dataset
Linear Regression	0.000002	0.0386
RNN	0.0043	0.0099
CNN	0.0056	0.0124

flat validation loss even after 20 epochs of training, while the CNN begins to overfit quickly. The CNN also seems to be very sensitive to initial conditions. Rerunning the final model with different random initialization produced significantly different results.

5.1 Aggregated Review Scores

The final step is to calculate an aggregated score based on the multiple user reviews for each of the movies in the test dataset. This was done by taking a simple average of all the scores for a given movie.

Table 2 shows the mean squared error for the aggregated review scores for the 219 movies in the test dataset. This MSE compares the averaged results of each models individual review predictions to the actual averaged review scores as given by the users in the original data.

As with the individual review score predictions, the RNN model performs best in predicting the average aggregated score.

5.2 Real-World Test Cases

In order to test the models on a real-word application, I selected six reviews from Rotten Tomatoes' top critics for four recent movies:

Table 2: Aggregated Review Score Mean Squared Error

Model	Test MSE
Linear Regression	0.0084
RNN	0.0013
CNN	0.0031

Joker [2], *21 Bridges* [1], *Knives Out* [3], and *Rambo: Last Blood* [4]. Reviews for *Joker* and *21 Bridges* were mixed, with 3 positive and 3 negative reviews each, resulting in a Rotten Tomatoes-style "sentiment score" of 0.5. The selected reviews for *Knives Out* were all positive, resulting in a sentiment score of 1.0 and the selected reviews for *Rambo: Last Blood* were all negative, resulting in a sentiment score of 0.

Table 3: Real-World Reviews Summary

Movie	Publication	Author	Sentiment Class
Joker	New York Times	A.O. Scott	Negative
	New Yorker	Richard Brody	Negative
	Slate	Dana Stevens	Negative
	Rolling Stone	Peter Travers	Positive
	Chicago Sun-Times	Richard Roeper	Positive
	Time Out	Phil de Semlyen	Positive
21 Bridges	Chicago Sun-Times	Richard Roeper	Negative
	Rolling Stone	Peter Travers	Negative
	Chicago Tribune	Michael Phillips	Negative
	San Francisco Chronicle	Mike LaSalle	Positive
	Washington Post	Michael O'Sullivan	Positive
	Time Out	Joseph Walsh	Positive
Knives Out	New York	David Edelstein	Positive
	Rolling Stone	Peter Travers	Positive
	The Atlantic	David Sims	Positive
	NPR	Linda Holmes	Positive
	The New York Times	Manohla Dargis	Positive
	Time	Stephanie Zacharek	Positive
Rambo: First Blood	New York	David Edelstein	Negative
	Time Out	Joshua Rothkopf	Negative
	Slate	Sam Adams	Negative
	AV Club	Ignatiy Vishnevetsky	Negative
	Variety	Peter Debruge	Negative
	San Francisco Chronicle	Peter Hartlaub	Negative

Figure 8 shows the predicted scores for each review based on the RNN model. Each symbol is coded by the movie being reviewed, and the location along the x-axis indicates whether the review was classified as positive or negative on Rotten Tomatoes. The horizontal black line indicates the mean normalized review score for the dataset used to train the RNN model.

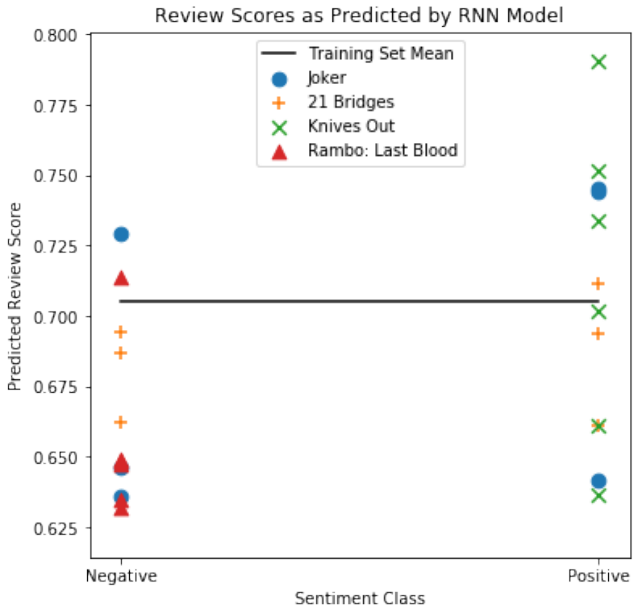

Figure 8: Predicted Review Scores for Real-World Reviews

Ten of the 12 negatively classified reviews have predicted scores less than the mean training score. This is an encouraging result. Unfortunately, the scores for the positively classified reviews are more scattered, with half above and half below the training data mean score.

Table 4: Aggregated Review Scores for the Four Recent Movies

Movie	Sentiment	Linear Regression	RNN	CNN
Joker	0.5	0.7095	0.6905	0.6208
21 Bridges	0.5	0.5986	0.6850	0.6318
Knives Out	1.0	0.7018	0.7125	0.6613
Rambo: Last Blood	0.0	0.6370	0.6543	0.6596

Table 4 shows the aggregated review scores for the four movies. The first column indicates the score based on the Rotten Tomatoes method, averaging the binary classification values. The next columns show aggregated scores based on the means of the predicted individual review scores from each of the three models.

Averaging the review scores to create the aggregate score seems to push each movie's score towards the mean score of the training data. This may be due to the skewed nature of the training data towards positive reviews or it may be a flaw in the methodology. It is hard to tell without further research/investigation.

6 CONCLUSIONS AND FUTURE WORK

While all three models were capable of predicting review scores, the RNN model is the clear winner in terms of lowest MSE on the test set for both the individual review scores and the aggregated scores. But even the RNN model had potential issues when applied to real-word movie reviews that were outside of the SNAP dataset. This may be due to a number of factors including the Amazon user reviews leaning more towards positive reviews or not being representative of the language that would be used in more formal professional critical reviews.

Based on the experimental results, I feel the RNN model has the most promise for future research. This model can be improved by using a larger training dataset. There are a number of ways to increase the training dataset size: 1) include more reviews from the SNAP dataset, 2) increase the vocabulary size from the 10,000 most common words in the training dataset to a larger number of words and 3) increase the number of words in the input sequence from 300 to a larger value. Options 1 and 2 would increase the training time but should increase the ability of the model to identify sequences that indicate good or bad reviews. The third option might introduce issues of "forgetting" information in the longer sequences. This issue, if it occurs, may be alleviated by applying self-attention to the sequence.

In addition, the human curators for Rotten Tomatoes produce representative quotes from each review as part of their manual review. This task can also be handled by NLP techniques, and an extractive text summarization model can be trained on the same dataset. A future implementation of this model could produce both a review score and summary quote for each individual review, along with the aggregated score for each movie.

REFERENCES

[1] 2019. 21 Bridges - Movie Reviews. https://www.rottentomatoes.com/m/21_bridges/reviews?type=top_critics Last accessed 9 December 2019.

[2] 2019. Joker - Movie Reviews. https://www.rottentomatoes.com/m/joker_2019/reviews?type=top_critics Last accessed 9 December 2019.

[3] 2019. Knives Out - Movie Reviews. https://www.rottentomatoes.com/m/knives_out/reviews?type=top_critics Last accessed 9 December 2019.

[4] 2019. Rambo: Last Blood - Movie Reviews. https://www.rottentomatoes.com/m/rambo_last_blood/reviews?type=top_critics Last accessed 9 December 2019.

[5] 2019. Rotten Tomatoes: About. https://www.rottentomatoes.com/about Last accessed 10 December 2019.

[6] Akshat Bakliwal et al. 2011. Towards Enhanced Opinion Classification using NLP Techniques. *Proceedings of the Workshop on Sentiment Analysis where AI meets Psychology (SAAIP)* (November 2011), 101–107. https://www.aclweb.org/anthology/W11-3715.pdf

[7] Richard Socher et al. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing* (October 2013), 1631–1642. https://www.aclweb.org/anthology/D13-1170.pdf

[8] Tomas Mikolov et al. 2013. Efficient Estimation of Word Representations in Vector Space. arXiv:cs.CL/1301.3781

[9] Karen Spärck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation* 28, 1 (1972), 11–21. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.115.8343&rep=rep1&type=pdf

[10] Daniel Jurafsky and James H. Martin. 2018. *Speech and Language Processing* (3 ed.). Chapter 6.

[11] Julian McAuley and Jure Leskovec. 2013. From Amateurs to Connoisseurs: Modeling the Evolution of User Expertise through Online Reviews. (2013). https://snap.stanford.edu/data/web-Movies.html Last accessed 27 October 2019.

[12] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? Sentiment Classification using Machine Learning Techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*. Association for Computational Linguistics, 79–86. https://doi.org/10.3115/1118693.1118704

[13] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. 1532–1543. http://www.aclweb.org/anthology/D14-1162