Movie Reviews Sentiment Analysis

Group 18 Final Project Report
ECS 170 Spring 2024

Tobias Cheung, Simona Kamat, Sarvesh Krishan,
Noel Lee, Megan Phan, Tiffany Su, Heidi Trinh

June 12, 2024

Github Repository:

`https://github.com/sarvesh-krishan/ecs170-final-project`

1 Introduction

The objective of this project was to develop a machine-learning model using Recurrent Neural Networks (RNNs) for classifying movie reviews as either positive or negative sentiment. The goal was to gain practical knowledge in building and optimizing RNNs, and to explore the potential applications of artificial intelligence in sentiment analysis within the domain of movie reviews. By leveraging the power of RNNs, this model can assist in a faster and more detailed analysis of large volumes of movie reviews, rather than relying on a traditional discrete rating system. Throughout the project, the main focus was on understanding the fundamentals of working with textual data and training RNN models on a dataset of labeled movie reviews, that after optimal hyperparameter tuning, achieved at least 70% accuracy, which we deemed as model performance indicative of a reliable and accurate sentiment classifier. By accomplishing this project, the team gained valuable insights into the potential applications of RNNs in natural language processing tasks, specifically sentiment analysis applied to other domains, such as product or restaurant reviews. With further refinements and adaptations, this RNN-based model could be deployed in real-world scenarios, assisting in sentiment analysis for movie reviews and supporting decision-making processes in the entertainment industry.

2 Background

2.1 Sentiment Analysis

To classify IMDB movie reviews as positive or negative, our project utilizes sentiment analysis. Sentiment analysis is used to extract subjective opinions about some material and is typically used to help businesses determine general response to a product (Gupta). Sentiment analysis can be used to extract subjective opinions from low-quality language data such as tweets. In "Sentiment Analysis: Concept, Analysis and Applications," an example of sentiment analysis was utilized to understand user opinions about Uber. They used data points from Facebook, Twitter, and the news to understand opinions about themes such as the payment, price, and safety of the business.

2.2 Natural Language Processing

Natural Language Processing (NLP) is a field of AI that uses machine learning to understand human language. NLP involves the training of machine learning models on text to determine correlations between words and a given output. While there are multiple methods of implementing NLP, we chose to implement it using a Recurrent Neural Network (RNN). At each time step, an RNN takes an input vector and the previous hidden state as input. The hidden state serves as the memory of the network, capturing information from previous time steps. The output of the recurrent unit becomes the hidden state for the next time step, and the process is repeated for the entire sequence. This allows the RNN to capture the temporal dependencies in the data, as information from previous time steps influences the computation at the current time step. As a result, the RNN model learns about patterns and associations between certain words, and how these are related to classifying the text as positive or negative sentiment. The RNN model can then be used to make predictions about the overall sentiment of a review.

2.3 Importance of Our Model

The ability to correctly interpret the sentiment of movie reviews can provide valuable insights into audience preferences and reactions, which can help in the decision-making process for businesses, movie producers, and viewers. Such feedback can help businesses decide what movies to recommend, movie producers how well a movie was received, and whether viewers searching for media to watch should watch a particular movie. Hence, models used to classify review sentiment as positive or negative help in automating the analysis of audience feedback.

3 Methodology

3.1 Dataset

The dataset used in this project is the Large Movie Review Dataset from Stanford. The dataset is curated for binary sentiment classification problems and consists of 50,000 labeled observations. We used the original 50:50 split and used 20% of the training data for validation during the training process. The distribution of the data is balanced with 25,000 negative reviews and 25,000 positive reviews. Reviews with a score of ≤ 4 out of 10 were classified as negative and reviews with a score of ≥ 7 were considered positive. Additionally, there was a maximum of 30 reviews for any given movie to reduce bias.

3.2 Exploratory Data Analysis

To better understand the text data that we were working with, we needed to generate a list of the top 10 most frequent words found in each class. Upon looking at the contents of the movie reviews, we noticed that there were many words that provided little semantic meaning and had little impact on the sentiment of the review itself. These words occurred frequently but would not provide an accurate representation of the words with significant meaning in the data. Therefore, we defined our own set of stop words to ignore in the data. This set consisted of common English stop words such as "a," "the," and "is" along with words specific to the data such as "movie," "scene," and "film." After removing the stop words, a list of the 10 most frequent words found was generated for either class, represented in Figure 1 and Figure 2.

Word	Frequency
good	7536
great	6381
well	5773
no	4454
best	4269
love	4219
show	3397
seen	3397
little	3325
still	3322

Figure 1: Top 10 frequent words in positive reviews

Word	Frequency
no	7966
good	7277
bad	7265
acting	3999
well	3929
did	3500
why	3440
being	3387
better	3341
know	3315

Figure 2: Top 10 frequent words in negative reviews

Furthermore, reviews from both positive and negative classes showed a wide range in the number of characters and words. The shortest review consisted of four words with the longest review having 2470 words. The median number of characters and words for either class were similar suggesting that the length of the review is not indicative of class. Negative reviews tend to be shorter than positive reviews at their maximum length, implying that negative feedback may be more concise or less detailed than positive feedback. The IQR for both classes is broad, however, the upper quartile in character count for positive reviews was slightly higher indicating a more detailed review when people prefer a movie. This supports the previous statement regarding concise negative reviews.

In addition to analyzing the content of the data, we also looked at the overall distribution of the data to better understand what we were using to train our model. Before any analysis, we had already known that neutral reviews (a score of 5 or 6) were not included in the dataset. After analyzing the distribution of the data, this was reinforced as we noticed that the data was heavily weighted towards either end of the rating scale. The histogram in Figure 3 shows that there were approximately 5000 observations that had a rating of 1, approximately 5000 observations that had a rating of 10, and less than 3000 occurrences of every other rating value between 2 and 9 inclusive. This indicates that the data is highly polar, allowing our model to achieve a better understanding of what is considered positive or negative and reducing ambiguity in the classification.

Statistic	Value
Count	25000
Mean	232.85
Std Dev	177.50
Min	10
25%	125
50%	172
75%	284
Max	2470

Table 1: Statistics for Words in Positive Reviews

Statistic	Value
Count	25000
Mean	1324.80
Std Dev	1031.49
Min	65
25%	691
50%	968
75%	1614
Max	13704

Table 2: Statistics for Characters in Positive Reviews

Statistic	Value
Count	25000
Mean	229.46
Std Dev	164.95
Min	4
25%	128
50%	174
75%	278
Max	1522

Table 3: Statistics for Words in Negative Reviews

Statistic	Value
Count	25000
Mean	1294.06
Std Dev	945.89
Min	32
25%	706
50%	973
75%	1567.25
Max	8969

Table 4: Statistics for Characters in Negative Reviews

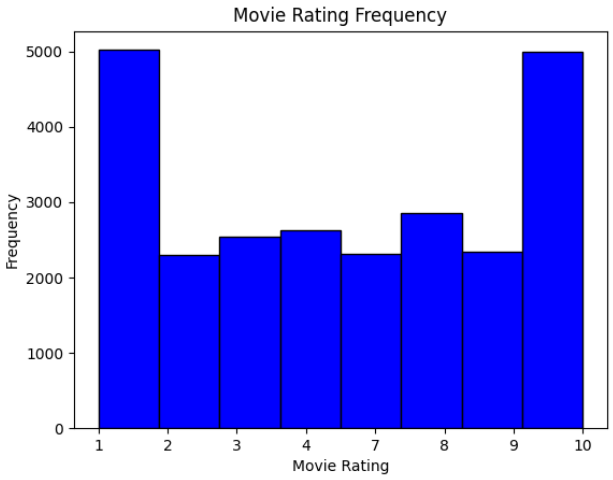

Figure 3: Distribution of Data by Rating Value

3.3 RNN

We employed a simple RNN for our initial model to serve as a baseline to compare with more advanced architectures we would implement later on. The architecture consisted of the following parts:

- The embedding layer consisted of 100 pre-trained GloVe word vectors. The indices of inputted words are converted into dense word vectors.
- The RNN layer processes the input, which consists of the input size and hidden layer size. The RNN layer processes the input sequences recurrently.

- The fully connected linear layer connects the final hidden state to the final category of sentiment categories consisting of positive and negative.

For further improvement of our initial model, we explored additional RNN architectures. We explored machine learning techniques such as LSTMs and GRUs to improve the robustness of our model. Written below are the different architectures utilized in our project:

3.4 LSTM

Long Short-Term Memory networks are a type of RNN that can overcome the vanishing gradient problem in traditional RNNs, wherein longer sequences suffer from increasingly smaller error signals from earlier text. LSTMs utilize memory cells and internal states in order for the gradient to pass steps without vanishing. Compared to RNNs, LSTMs are also useful for learning long-term dependencies in data and are well-suited for tasks of sequential data. The architecture consisted of the following parts:

- The input layer, which received the dimension 100 GloVe embeddings for each word in the review. The input to LSTM was batches of sequences represented as a matrix.
- The LSTM layer, a layer with 128 hidden units that processed the input sequences and produced output to capture spatial dependencies and contextual information. It was also configured to be bidirectional, so the model could look both forward and backward to better capture the context in the text data.
- The fully connected dense later, which maps the outputs to the final category classes of positive vs negative.

3.5 GRU

Gated Recurrent Unit networks (GRUs) are a simpler form of LSTMs that can overcome the vanishing gradient problem by using a single gate to control information flow into the memory cell[3]. It allows for fewer parameters, making GRUs computationally more efficient. Its simplicity compared to LSTM can lead to faster training times while still capturing essential dependencies in the data[3]. In our project, we implemented a GRU to balance between model complexity and training efficiency. The architecture consisted of the following parts:

- In the embedding layer, our model used 100 pre-trained GloVe vectors for word embeddings.
- The GRU layer processes the input, which consists of our input size and the number of units in the hidden layer. We experimented with two different hidden layer sizes, 128 and 256. We extended our GRU with bidirectional analysis, allowing the model to grasp the context by taking the words before and after each word.
- After running through the input sequences, the fully connected layer transforms the final hidden state to output a result of either positive or negative from the sentiment categories.

3.6 Hyperparameters

For each type of architecture implemented, we chose to tune different hyperparameters in order to improve performance. Within each architecture, we altered the learning rate, hidden layer size, and optimizer. Each model created utilized a unique combination of each hyperparameter tuning. This created 12 models for each architecture implemented and 36 models overall. Below are the specific hyperparameter values experimented with:

- Learning rates: 0.0001, 0.001, and 0.01
- Hidden layer size: 128 and 256
- Optimizer: Adam and SGD (Stochastic Gradient Descent)

4 Results

4.1 Initial RNN Model

The initial Recurrent Neural Network (RNN) model achieved an evaluation accuracy on the testing set of 52.36% after taking 38.53 seconds to train the model and taking 5.61 seconds to evaluate the model. The precision value of 0.52 was significantly less than the recall value of 0.75, which suggests that the model is better at correctly identifying all positive movie reviews (minimizing false negatives) rather than accurately classifying positive movie reviews (minimizing false positives). From the confusion matrix, we see that the initial RNN model had 8812 false positives, much higher than the 3099 false negatives, and it predicted observations as positive movie reviews. It could be that the model fails to identify the negative connotation of words when grouped together i.e. "This movie is not very good."

Initial RNN Confusion Matrix		
	Predicted Negative	Predicted Positive
True Negative	3688	8812
True Positive	3099	9401

Table 5: Confusion Matrix for Initial RNN Model

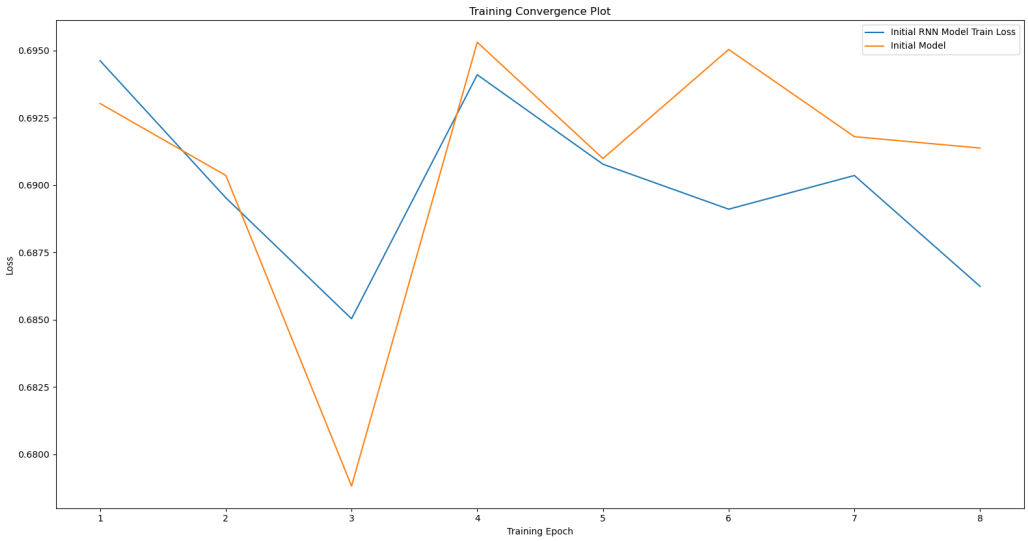

Figure 4: Training Convergence Plot for Initial RNN Model

The training convergence plot of the initial RNN model suggests training was sporadic at best with the training and validation loss jaggedly decreasing. The magnitude of loss decrease was not significant either, leading us to believe the model trained very poorly on the training and validation data sets. The Receiver Operating Characteristic (ROC) curve supports this analysis as the initial RNN model barely performed better than a random chance classifier.

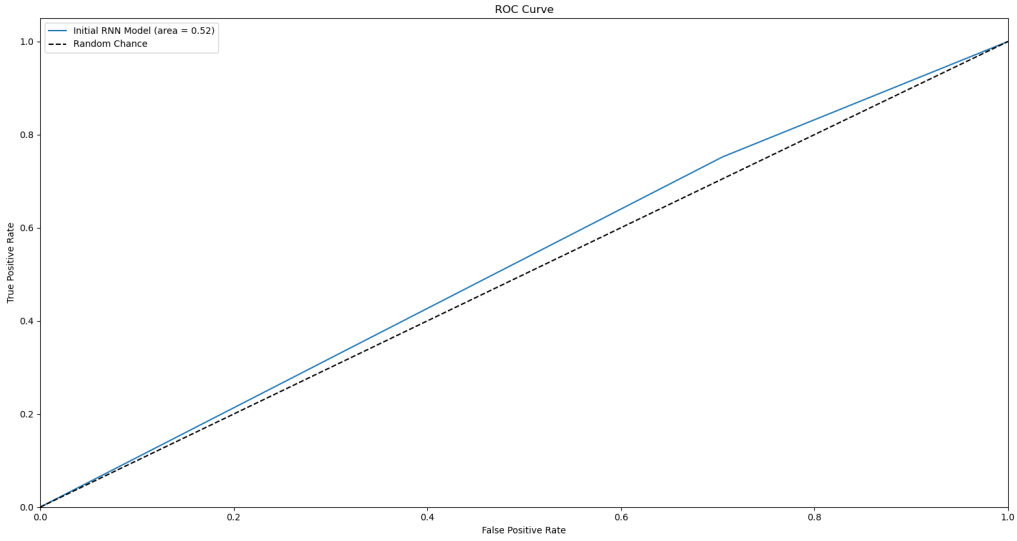

Figure 5: Receiver Operating Characteristic Plot for Initial RNN Model

4.2 Hyperparameter Tuning & Model Selection

After thirty-six (36) models were trained on the training and validation data sets and evaluated on the testing data set, the performance metrics of each model were recorded. The twelve (12) RNN models achieved a mean evaluation accuracy of 50.76% with a standard deviation of 0.83% after training for an average of 33.5 seconds with a 3.5-second standard deviation. This suggests that the twelve (12) RNN models did not perform significantly differently from one another despite having different hyperparameters. Regardless of the hyperparameter values, the base RNN architecture was unable to discern textual data and classify movie reviews into positive and negative sentiments. The twelve (12) LSTM models achieved a mean evaluation accuracy of 65.15% with a standard deviation of 15.30% after training for an average of 60.9 seconds with a 24.4-second standard deviation. The LSTM models performed distinctly from one another as seen by the larger standard deviation of the accuracy and the training time with the 6 models using Adam as their optimizer performing significantly better than the 6 models using Stochastic Gradient Descent as their optimizer. The LSTM models with SGD as their optimizer also typically had significantly shorter training times with a 40.7-second average as these models incurred early stopping when the validation loss had plateaued. The LSTM models with Adam as their optimizer obtained a minimum accuracy of 75.4% and reached a maximum accuracy of 83.9% over an average training time of 81.1 seconds. Model #21 is the most accurate LSTM model, which had a hidden layer size of 128 and a learning rate of 0.01.

The training convergence plot of the LSTM Set 3 models with a learning rate set to 0.01 shows the Adam models (Model #21 & #23) training reasonably well with training loss decreasing and validation loss plateauing, which led to the early stop at 10 epochs. Unfortunately, the LSTM models (#22 and #24) with SGD trained very poorly as both training and validation loss stayed constant until terminating at 6 epochs. The ROC curve supports these findings as the two Adam LSTM models are concave towards the top left, indicating a higher TPR and lower FPR, which translates to better overall performance in classification whereas the two SGD LSTM models are close to the Random Chance classifier line, suggesting that they are not much better than randomly classifying the reviews.

Model #21's confusion matrix (Table 6) shows that it predicted slightly more accurately for negative movie reviews relative to positive movie reviews. With an accuracy of 83.9%, this LSTM model certainly contends as the best model to classify our movie reviews and showcases the strength of the LSTM architecture compared to the base RNN when capturing patterns within textual data.

The twelve (12) GRU models achieved a mean evaluation accuracy of 63.53% with a standard deviation

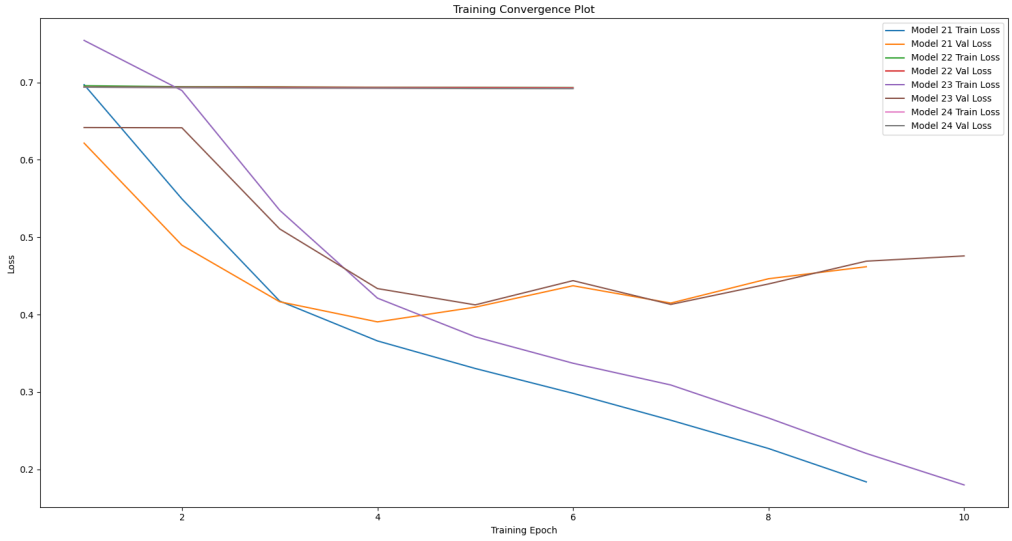

Figure 6: Training Convergence Plot for LSTM Set 3 Models

Model #21 Confusion Matrix		
	Predicted Negative	Predicted Positive
True Negative	10829	1671
True Positive	2869	9631

Table 6: Confusion Matrix for Model #21

of 15.27% after training for an average of 48.7 seconds with a 20.7-second standard deviation. Similar to the LSTM models, the 6 GRU models using Adam as their optimizer performed significantly better than the 6 models using Stochastic Gradient Descent as their optimizer. The GRU models with SGD as their optimizer also typically had significantly shorter training times with a 35.3-second average as these models incurred early stopping when the validation loss had plateaued. The LSTM models with Adam as their optimizer obtained a minimum accuracy of 53.8% and reached a maximum accuracy of 84.1% over an average training time of 62.1 seconds. Model #33 and Model #35 are the most accurate GRU models, both of which had a learning rate of 0.01 and a hidden layer size of 128 and 256, respectively. Note that Model #29 also performed exceptionally well with an evaluation accuracy of 83.7% but was not selected with Models #33 and #35 as its training time of 78.3 seconds was significantly greater than the other two models.

The training convergence plot of the GRU Set 3 models with a learning rate set to 0.01 shows the Adam models (Model #33 & #35) training reasonably well with training loss decreasing and validation loss slightly increasing, which led to the early stop at 9 and 10 epochs. Unfortunately, the GRU models (#34 and #36) with SGD trained very poorly as both training and validation loss stayed constant until terminating at 6 epochs. The ROC curve supports these findings as the two Adam GRU models are concave towards the top left, indicating a higher TPR and lower FPR, which translates to better overall performance in classification whereas the two SGD GRU models are close to the Random Chance classifier line, suggesting that they are not much better than randomly classifying the reviews. This is the same phenomenon that was observed with the LSTM models.

Similar to LSTM Model #21, Model #33's confusion matrix (Table 7) shows that it predicted slightly more accurately for negative movie reviews relative to positive movie reviews.

Contrary to LSTM Model #21 and GRU Model #33, Model #35's confusion matrix (Table 8) shows that it predicted slightly more accurately for positive movie reviews relative to negative movie reviews. Regardless,

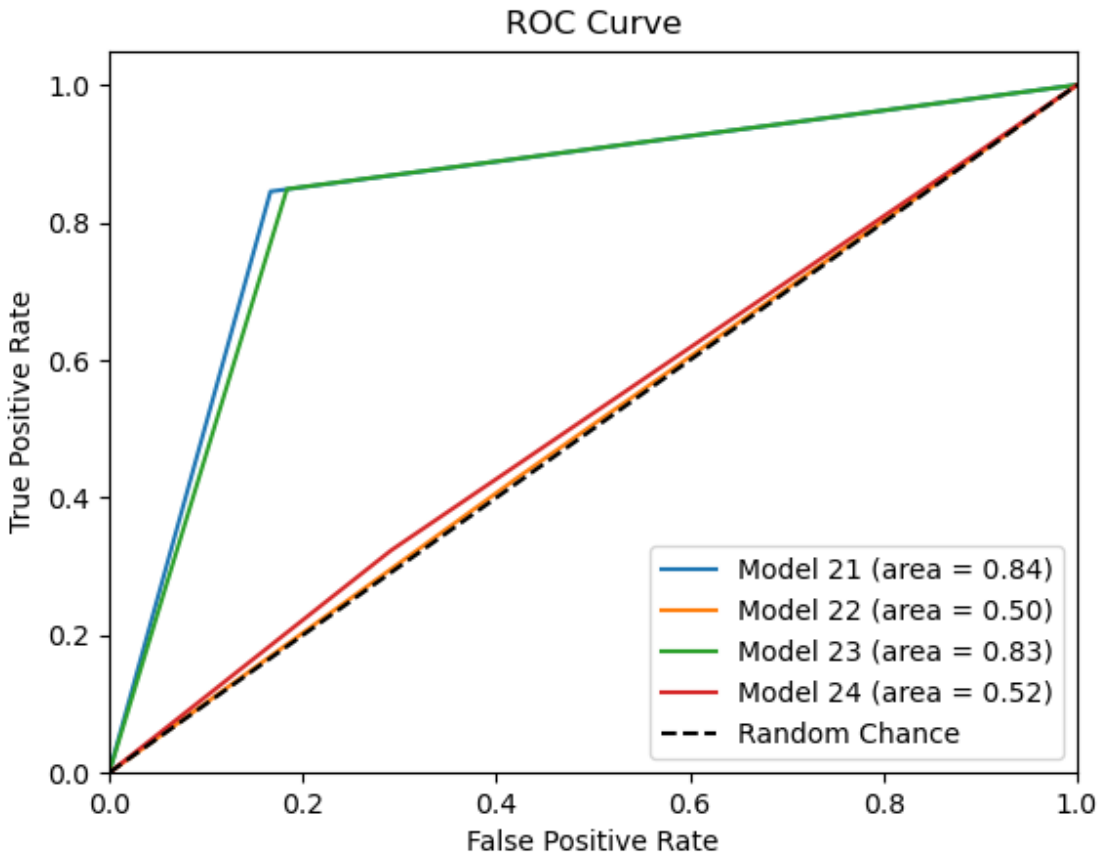

Figure 7: Receiver Operating Characteristic Plot for LSTM Set 3 Models

Model #33 Confusion Matrix		
	Predicted Negative	Predicted Positive
True Negative	11186	1314
True Positive	2840	9660

Table 7: Confusion Matrix for Model #33

Model #35 Confusion Matrix		
	Predicted Negative	Predicted Positive
True Negative	9100	3400
True Positive	860	11640

Table 8: Confusion Matrix for Model #35

both Models #33 and #35 perform exceptionally well within the GRU models set and are joining Model #21 to form the optimal model set.

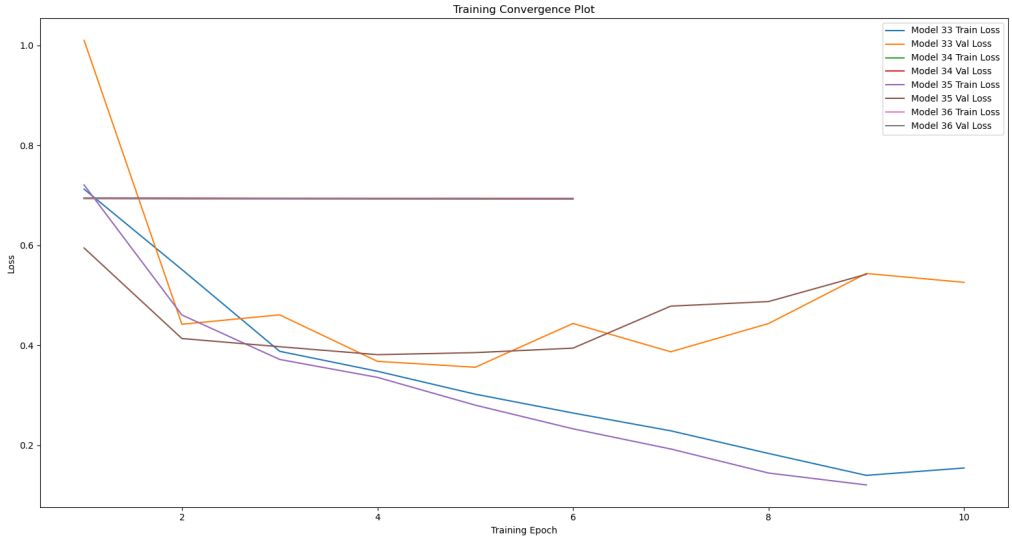

Figure 8: Training Convergence Plot for GRU Set 3 Models

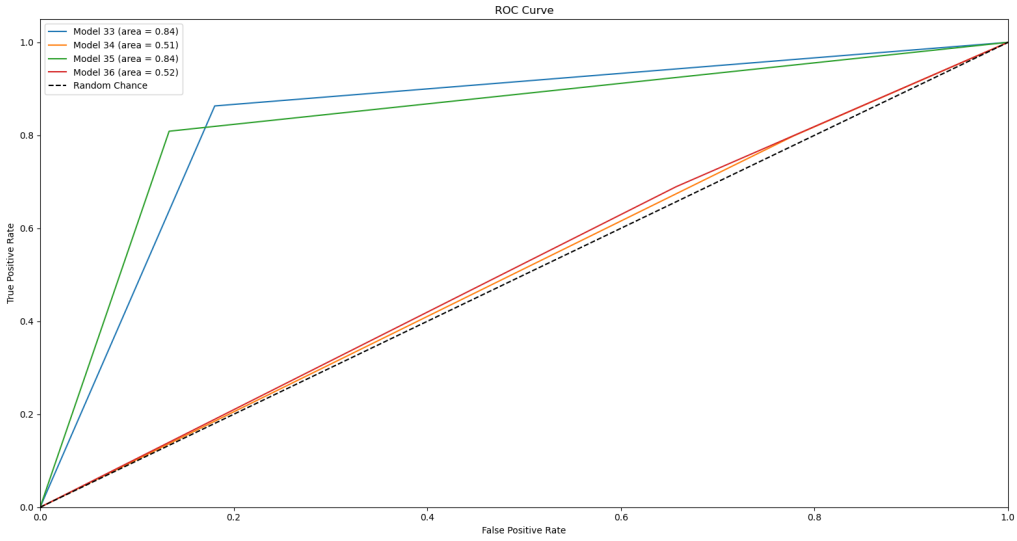

Figure 9: Receiver Operating Characteristic Plot for GRU Set 3 Models

4.3 The Optimal Models

The training convergence plot of the three optimal models shows the models all training reasonably well with training loss decreasing and validation loss slightly increasing, which led to the early stops at 8, 9, and 10 epochs. Model #33 had the lowest training loss but Model #21 had the lowest validation loss. We will lean towards Model #21 being the final optimal model as it is generally desirable to choose the model with lower validation loss, even if it has a higher training loss. The validation loss is a more reliable indicator of a model's performance on unseen data, which is ultimately the goal of our final model. Model #21's lower

validation loss suggests better generalization and indicates that the model is capturing meaningful patterns rather than memorizing the training data. We can also infer that Model #33's significantly lower training loss but higher validation loss may be a sign of overfitting. The ROC curves show all three models having similar but excellent AUC values of approximately 0.83. However, Model #35's curve is higher but further to the right, suggesting that the classifier is more sensitive to positive movie reviews, even at the cost of a slightly higher False Positive Rate (FPR). The curves of the other two models are closer to the left (lower FPR) but lower on the plot, suggesting that the classifiers maintain a lower rate of false positives, even if they sacrifice some true positive classifications. When looking at training runtimes, Model #21 is the slowest at 52.3 seconds, followed by Model #33 at 51.1 seconds, and the fastest training model is Model #35 at 47.3 seconds. Note that Model #35 stopped earlier than Model #33, which we attribute to the run-time difference, despite expecting larger hidden layer size models to take longer to run. This leads us to conclude that Model #21 is our final optimal model using the LSTM architecture, a learning rate of 0.01, the Adam optimizer, and the hidden layer size set to 128. We have achieved our goal of building and optimizing an RNN model to accurately predict the sentiment class of movie reviews.

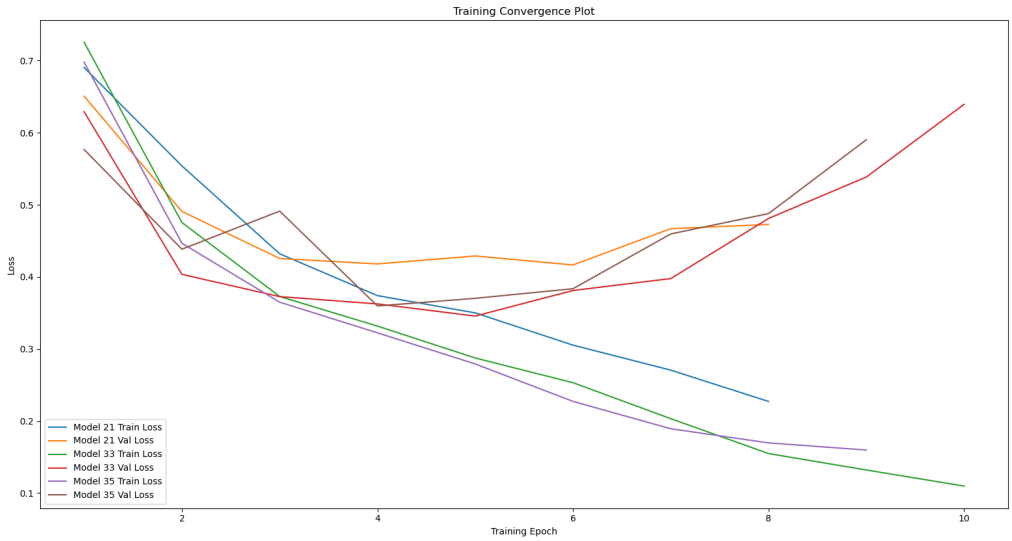

Figure 10: Training Convergence Plot for Top 3 Models

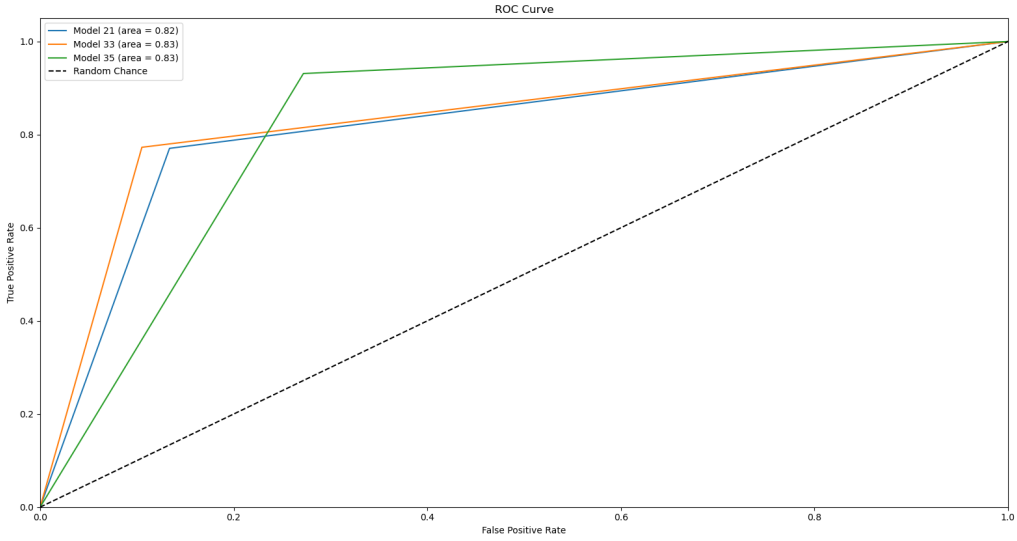

Figure 11: Receiver Operating Characteristic Plot for Top 3 Models

5 Discussion

There are some general takeaways we have from the training and evaluation data of 36 models. The base RNN model performed poorly regardless of how we changed hyperparameter values. Only incorporating information from past time steps through the hidden state as well as the vanishing gradient problem severely limits the ability of the base RNN model to capture long-term dependencies and correctly classify textual data. The usage of LSTM and GRU architectures that introduce gating mechanisms allows networks to control the flow of information, which solves the vanishing gradient problem and helps the network capture long-term dependencies in sequences. As a result, the LSTM and GRU models far outperform base RNN models and many of these models achieve the 70% accuracy goal we set.

Despite some models doubling the hidden layer size, the accuracy did not increase as much as we expected. We infer that a hidden layer size of 128 captures most of the meaningful patterns needed to accurately classify our movie reviews and increasing the size of the hidden layer did not capture more meaningful patterns. However, the models with double the hidden layer size generally had increased training times since the hidden layer size directly impacts the computational complexity of the neural network. Increasing the hidden layer size adds more parameters to the network, resulting in more computations during forward and backward propagation. As a result, larger hidden layer sizes can lead to longer training times due to the increased computational workload. Increasing the learning rate generally resulted in decreased training times as the larger step size allows for larger parameter updates, which can lead to faster convergence.

6 Conclusion

In our initial proposal, we planned to build a simple RNN model, iteratively tune the hyperparameters, and evaluate the resulting models against previous iterations until we reached an approximate testing accuracy of 70%. In the end, we surpassed this goal, achieving a testing accuracy of above 80% in multiple models. We accomplished this by utilizing LSTM and GRU architectures and finding optimal hyperparameters for the models.

To further refine our model for the best possible performance, we can run the better-performing models

multiple times and collect average evaluation metrics. We can use this data in hypothesis testing and determine statistically significant differences. Finally, we can test our optimal model on data from other disciplines to observe how our model generalizes to reviews outside of movies, such as restaurant review data or opinions on Twitter.

7 Contribution

7.1 Tobias

Created rating distribution histogram, tested initial model, tuned hyperparameters (optimizer, hidden layer size, learning rate, and architecture) for all thirty-six models. Collected data and plots for each model. Assisted in the Results section.

7.2 Simona

Implemented run-time calculation of models, implemented validation set and its loss, and assisted with final project report.

7.3 Sarvesh

Planned and led meetings to inform the team of the project's status & issues, assigned tasks to the team and assisted them where needed, created and maintained the GitHub repository, implemented the model classes and their train and evaluate methods for RNN, LSTM, GRU, the loading, processing and batching of data to feed into the models, and the code template to instantiate, tune, train and evaluate a model. Wrote the Introduction, Results, and Discussion sections of the final report.

7.4 Noel

Calculated word frequencies by class. Assisted on the LSTM subsection of the Methodology section of the project report.

7.5 Megan

Calculated five-number summary statistics on word and character length of reviews by class and created presentation slides

7.6 Tiffany

Plotted the distribution of review ratings per class for the dataset, worked on the Background and References section for the written report.

7.7 Heidi

Implemented the training convergence plot of models and the early stopping mechanism to prevent model over-fitting. Worked on portions of the Methodology section, and the Conclusion section and assisted in other sections of the final report.

References

[1] Gupta, Shashank. "Sentiment Analysis: Concept, Analysis and Applications." *Medium*, Towards Data Science, 19 Jan. 2018, towardsdatascience.com/sentiment-analysis-concept-analysis-and-applications-6c94d6f58c17.

[2] Holdsworth, Jim. "What Is NLP (Natural Language Processing)?" *IBM*, IBM, 23 Sept. 2021, www.ibm.com/topics/natural-language-processing.

[3] Kostadinov, Simeon. "Understanding GRU Networks." *Medium*, Towards Data Science, 16 Dec. 2017, towardsdatascience.com/understanding-gru-networks-2ef37df6c9be.

[4] Maas, Andrew L., et al. 'Learning Word Vectors for Sentiment Analysis'. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, edited by Dekang Lin et al., Association for Computational Linguistics, 2011, pp. 142–150, aclanthology.org/P11-1015.

[5] Pennington, Jeffrey, et al. *Glove: Global Vectors for Word Representation*, Stanford NLP Group, 2014, nlp.stanford.edu/projects/glove/.

8 Appendix: Google Sheet and Google Drive

8.1 Model Data - Google Sheets

The Google Sheet used in this document can be accessed using the following link:

`https://docs.google.com/spreadsheets/d/10brnTFktcT7rwd0X3TJ0H9OQo9kzRvJhEJRyDTNCNQI/edit?gid=0#gid=05`

This spreadsheet contains all the data collected for each of the 36 models and the initial RNN model.

8.2 Model Plots - Google Drive

All TCP and ROC plots gathered for each model are found in the Google Drive folder linked below.

`https://drive.google.com/drive/folders/1O7Y1KryiIu8uOmCcgCcrr_mPYRtCzceY?usp=sharing`