Sentiment Analysis using Logistic Regression and Neural Networks

Abhinav Umat

Department of Computer Science, Wright State University

Email:- umat.2@wright.edu

Abstract

Sentiment analysis is a Natural Language Processing Technique, which is used to determine whether a piece of writing is positive, negative or neutral. It combines varies natural language processing techniques along with machine learning techniques to achieve the goal. In this paper I will discuss how different Machine Learning Classification Algorithms, such as logistic regression and Neural Networks perform in classifying whether a piece of movie review is either positive or negative. For this project I have built four different models, one model using logistic regression and three models using Neural Networks and Compared between them which one performed the best.

1. Introduction

Sentiment analysis is an area of Machine Learning which is used quite extensively by large enterprises. It is used in gathering public opinion, monitor brand reputation, which in turns helps in understanding customer experience. Big companies such as Facebook, Twitter, Amazon to smaller companies all use sentiment analysis to understand how a customer or an employee feels about a related topic and why they feel like this. This data can really help them make improvement to their products and services to help improve customer and employee experience.

For this project I wanted to experiment with different Machine learning techniques and how well they perform in classifying whether a movie review is positive or negative. I have built four models, one logistic regression model, one multilayer perceptron Neural Network using Stochastic gradient descent (SGD) optimizer, one multilayer perceptron Neural Network using Adam optimizer, and one Recurrent Neural Network (RNN's) using Adam optimizer.

I divided my dataset into Training and Testing set. I used a 70-30 approach. I randomly selected 70% of the data for the training set and 30% of the data for the testing set. After preprocessing of my text and diving the dataset into training and testing sets, I used Scikit-learns logistic regression model for training and testing. I got 51.4% test accuracy. Then I used Keras library for building my Neural Networks and trained and tested them on the training and testing dataset respectively.

The goal of this project is to find out which model performs the best classification task for classifying text documents in this case movie reviews and use that model in predicting whether a review is positive or negative. Recurrent Neural Networks with Adam optimizer worked the best for this dataset with 82.9% Test Accuracy.

2. Related Research

There are plenty of research which is going on in the area of sentiment analysis. Some of those researches I have listed below:

- **Preprocessing Techniques and Data Augmentation for Sentiment analysis by Huu-Thanh Duong and Tram-Anh Nguyen-Thi.**

  In the paper, they have used semi- supervised learning and data augmentation techniques to achieve the goal of sentiment analysis, which is really good for limited datasets and does a lot better on unseen data. In the paper they talk about how these techniques do really well on unstructured sentiments which are informal and free text [1].

- **Sentiment analysis on Adventure Movie scripts by Rifat Rahman, Md Abdul Masud, Raonak Jahan Mimi, and Mst. Nusrat Sultana Dina.**

  In the paper, they have collected a dataset consisting of 20 adventure genre movie scripts for sentiment analysis. They have used Minimum Viable Product (MVP), Exploratory Data Analysis (EDA), and

TextBlob sentiment analysis techniques to achieve their goal. Their results show that there are fluctuations of the sentiment that the adventure movie genre reflect the viewer's mindset as public opinion for social impact [2].

- **Semantic sentiment analysis of twitter by Hassan Saif, Yulan He, and Harith Alani.**

  In the paper, they talk about how we can add semantics as an additional feature in the training set for sentiment analysis. For example, they have for each extracted entity (e.g., iPhone) from tweets, added its semantic concept (e.g., "Apple product") as an additional feature, and measure the correlation of the representative concept with negative/positive sentiment. They have used three different twitter datasets for the experiments, Stanford Twitter Sentiment Corpus (STS), Health Care Reform (HCR), and Obama-McCain Debate (OMD) [3].

- **Sentiment Analysis of Movie Review Comments by Kuat Yessenov and Sasa Misailovi ˇ c.**

  In the Paper, they have used movie review comments from popular social network Digg as their dataset and classify text by subjectivity/objectivity and negative/positive attitude. They have used bag-of-words model by restricting to adjectives and adverbs, handling negations, and bounding word frequencies by a threshold, for extracting text features. And they have used various machine learning techniques such as Naive Bayes, Decision Trees, Maximum-Entropy, and K-Means clustering for evaluating accuracy [4].

3. Exploratory data analysis

For this project, I have about 49,970 IMDB movie reviews with positive or negative reviews. There are about 24,982 positive reviews and 24,987 negative reviews. Both classes are fairly balanced and even, one class does not take precedent over the other [5].

Figure 1 shows the number of samples I have for both the classes positive and negative; it shows that they are fairly balanced, there is no class imbalance for this dataset. All of the data is in text format for which I have used various preprocessing techniques to prepare them for the models that I have built to perform sentiment analysis.

There are plenty of samples with which I can divide my data into training and testing datasets. I have used 70-30 approach for dividing my dataset into training and testing sets respectively.

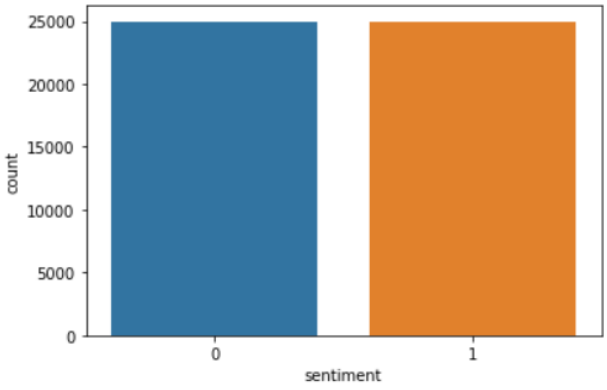

Fig. 1. Count of Observations for both Classes

4. Methods

Preprocessing Steps

There are plenty of preprocessing techniques that we can use to prepare our text data for data analysis. The steps that I used for preprocessing are as follows:

- First, I converted all of the reviews into lowercase, so that all of the reviews will be in same case.
- I then removed all the punctuations from all the reviews. I used string library of python which consists of some pre-defined list of punctuations such as '!" #$%&'()*+,-./:;?@[\]^_`{|}~' .
- Then I used python library re for the tokenization of the words, in which text is split into smaller units.
- Lastly, I used python library Porter Stemmer for stemming the words for all the reviews. It is a technique where words are stemmed or diminished to their root form.

After preprocessing of my text data, for encoding of the text I used TF-IDF (Term frequency – Inverse document frequency) technique. It is a technique which is intended to reflect how important a word is to a document or in this case a movie review.

I then divided my dataset into training set and testing set. I randomly selected 70% of the data for the training set and 30% of the data for the testing set. I also shuffled the entire dataset before dividing it into train and test set. I then padded the sequences for both training set and testing set to ensure all the sequences are of the same length. I have also used 20% of the 70% of training set as validation set while training. After I finished my preprocessing steps, I went on to build my models and train and test them.

Parameters for my Models

I built and trained in total of four models, one model using logistic regression and three models using Neural Networks. For the logistic regression model, I used the Scikit-learns logistic regression model for training and testing. It gave me 51.4% test accuracy which is not good. I then built two Multilayer perceptron Neural Network (MLP) using Keras library of python, one with Stochastic gradient descent (SGD) optimizer, one multilayer perceptron Neural Network using Adam optimizer.

Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, 100, 64)	9612736
flatten (Flatten)	(None, 6400)	0
dense (Dense)	(None, 50)	320050
dense_1 (Dense)	(None, 20)	1020
dense_2 (Dense)	(None, 10)	210
dense_3 (Dense)	(None, 2)	22

Total params: 9,934,038
Trainable params: 9,934,038
Non-trainable params: 0

Fig. 2. Final architecture for the MLP

Figure 2 shows my final architecture for the Multilayer perceptron Neural network, I used 1 embedding layer, then I converted the matrix into vectors by using flattening them, then I used in total of 3 hidden layers, with 50, 20, and 10 hidden nodes respectively. I also used one hot encoding for my target values. I played around with different structures beginning with 1 hidden layer and making my way to 3 hidden layers. But this structure gave the best results so, I picked this one. I also used sigmoid activation function for all of my layers. I also played around with different activation, such as relu, tanh, I also removed activations from certain layers to see which one performed best. Sigmoid activation seems to worked best when applied to all the layers. I think if we are dealing with only two classes, sigmoid seems to work best. I set my learning rate to be 0.01. I used binary cross entropy, since we are only dealing with two classes. I trained my neural network for 10 epochs.

MLP with Stochastic gradient descent optimizer gave me 50% test accuracy and MLP with Adam optimizer game me 81.8% test accuracy. Certainly, Adam optimizer worked the best for this dataset, so I built another Neural Network this time using Recurrent neural Networks (RNN's) using the same Adam optimizer.

Layer (type)	Output Shape	Param #
embedding_5 (Embedding)	(None, 100, 64)	9612736
lstm_5 (LSTM)	(None, 64)	33024
dense_13 (Dense)	(None, 50)	3250
dense_14 (Dense)	(None, 20)	1020
dense_15 (Dense)	(None, 10)	210
dense_16 (Dense)	(None, 2)	22

Total params: 9,650,262
Trainable params: 9,650,262
Non-trainable params: 0

Fig. 3. Final architecture for the RNN

Figure 3 shows my final architecture for the Recurrent Neural network, for the RNN I pretty much used the same structure but added 1 LSTM (Long short-term Memory) Layer with 64 units. For activation I used sigmoid and set my learning rate to be 0.01. I used binary cross entropy, since we are only dealing with two classes. I trained my neural network for 10 epochs. My test accuracy with this model is 82.9% which is slightly better than MLP with Adam optimizer. I played around with the structure of the model to see if I can achieve a better accuracy maybe even more than 90%, but this is the best I could get. My test accuracy ranged in between early 80's. I think one of the reasons this is because dataset is quite noisy. It's divided between two classes, positive or negative but a lot of those reviews might be neutral, a lot of people have mixed opinions about a lot of movies, they don't hate it that much or like it that much. I think if there were three classes, positive, negative and neutral then model might have performed much better.

5. Results

Below are the results for the four different models that I built:

Logistic Regression Model

Figure 4 shows the confusion matrix for the logistic regression model. The test accuracy for this model is 51.4%. It is not that good. It does do a slightly better job for the positive class as compared to the negative class, but results are still terrible.

Figure 5 shows the Classification report for the logistic regression model and how well it did on both the classes. It shows, precision for class 0 is 0.52 and class 1 is 0.51. There is not much difference between them. Recall for class 0 is 0.41 and class 1 is 0.62, which is much better than class 0. F1 score for class 1 which is 0.56 compared with 0.46 for the class 0 is also much better. All in all, this model performed slightly better on positive class when compared to the negative class but overall, the performance of the model is not that good.

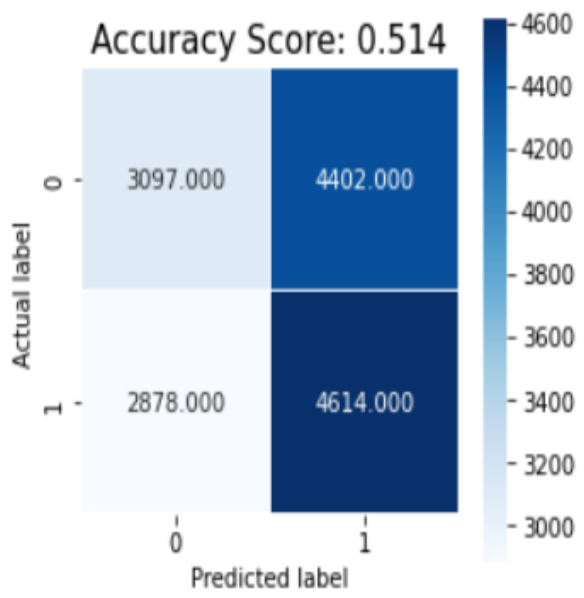

Fig. 4. Confusion Matrix for the Logistic Regression Model

	precision	recall	f1-score	support
0	0.52	0.41	0.46	7499
1	0.51	0.62	0.56	7492
accuracy			0.51	14991
macro avg	0.52	0.51	0.51	14991
weighted avg	0.52	0.51	0.51	14991

Fig. 5. Classification report for the Logistic Regression Model

Multilayer perceptron Neural Network with SGD optimizer

Figure 6 and Figure 7 shows the Training and Validation accuracy, and Training and validation loss respectively for 10 epochs for the Multilayer perceptron Neural network with SGD optimizer. According to the graph you can see that at about epoch 5 training starts to get a bit noisy. Also, for the loss there is a significant decrease initially but then it gets constant.

Figure 8 and Figure 9 shows the Confusion Matrix and Classification report respectively for this model. Although test accuracy for this model is 50% which is pretty similar to the logistic regression model but I believe this model has performed much worse than logistic regression model. As you can see in the classification report, it absolutely fails to predict the negative class. All the precision, recall and F1 score are 0. It only manages to predict the positive class which I don't think is a good sign. Even with pretty much similar test accuracy's I believe logistic regression model performed much better than this model.

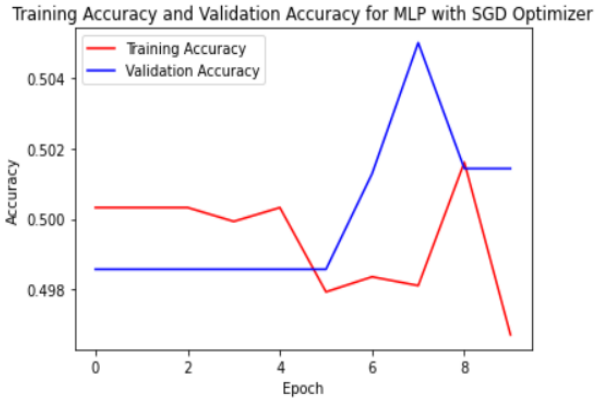

Fig. 6. Training and validation Accuracy for the MLP with SGD optimizer Model

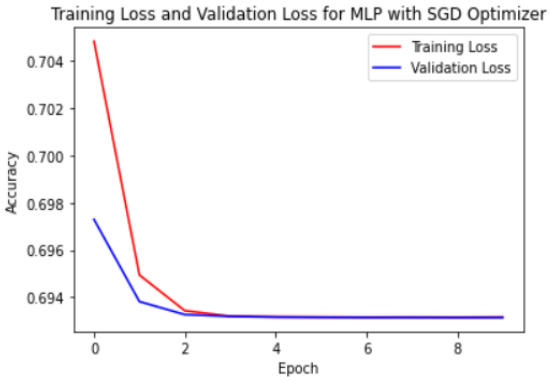

Fig. 7. Training and validation Loss for the MLP with SGD optimizer Model

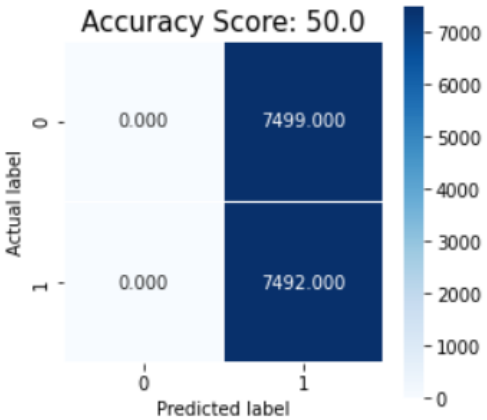

Fig. 8. Confusion Matrix for the MLP with SGD optimizer Model

	precision	recall	f1-score	support
0	0.00	0.00	0.00	7499
1	0.50	1.00	0.67	7492
accuracy			0.50	14991
macro avg	0.25	0.50	0.33	14991
weighted avg	0.25	0.50	0.33	14991

Fig. 9. Classification report for the MLP with SGD optimizer Model

Multilayer perceptron with Adam Optimizer

Figure 10 and Figure 11 shows the Training and Validation accuracy, and Training and validation loss respectively for 10 epochs for the Multilayer perceptron Neural network with Adam optimizer. As you can see training accuracy does keep increasing after every epoch but the validation accuracy stays pretty much the same. Though it is higher than when compared with the SGD optimizer model but it still remains pretty much the same. And as for the loss the validations loss keeps increasing after every epoch.

Figure 12 and Figure 13 shows the Confusion Matrix and Classification report respectively for this model. Test accuracy for this model is 81.8% which is much better than previous 2 models. Precision, recall, and F1 score for the class 0 are 0.83, 0.81, 0.82 respectively. And precision, recall, and F1 score for the class 1 are 0.81, 0.83, 0.82 respectively. According to these values I can say this model performed well on both the classes. Even though the training process was not that good this model still does a pretty good job. Adam optimizer seems to have performed much better for this dataset when compared with SGD optimizer.

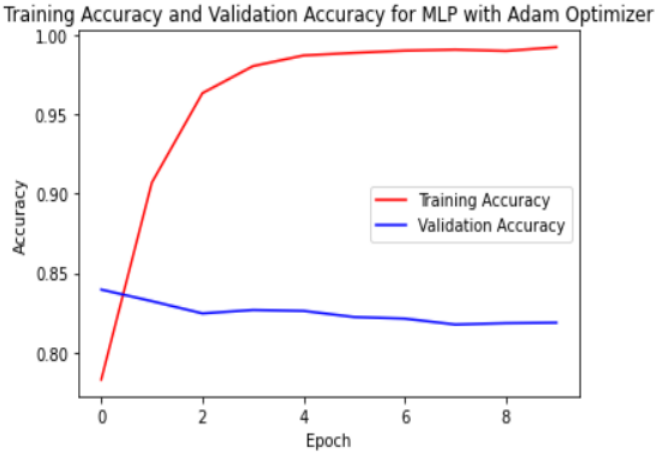

Fig. 10. Training and validation Accuracy for the MLP with Adam optimizer Model

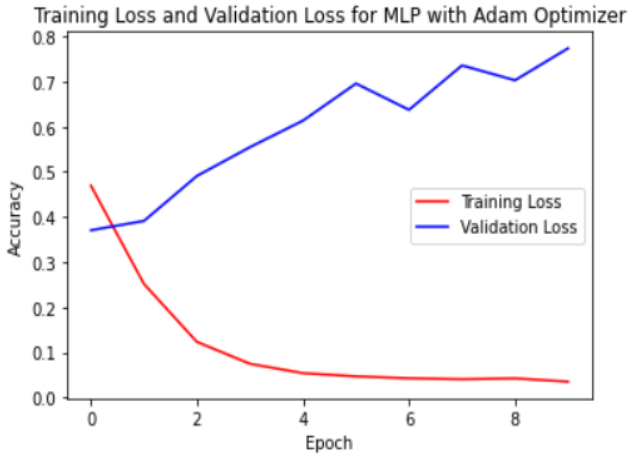

Fig. 11. Training and validation Loss for the MLP with Adam optimizer Model

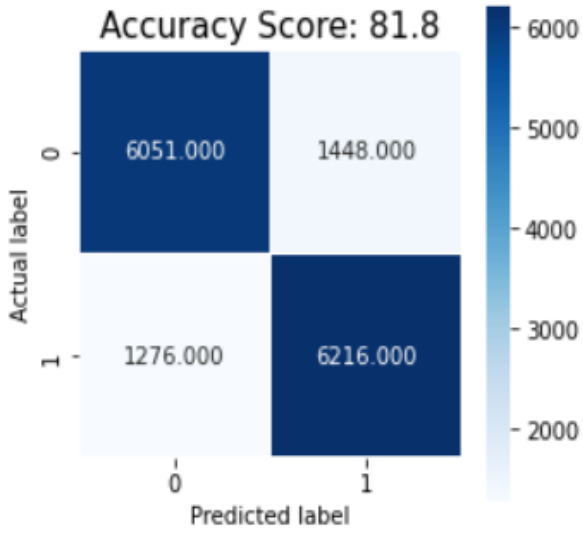

Fig. 12. Confusion Matrix for the MLP with Adam optimizer Model

	precision	recall	f1-score	support
0	0.83	0.81	0.82	7499
1	0.81	0.83	0.82	7492
accuracy			0.82	14991
macro avg	0.82	0.82	0.82	14991
weighted avg	0.82	0.82	0.82	14991

Fig. 13. Classification report for the MLP with Adam optimizer Model

Recurrent Neural network with Adam Optimizer

Figure 14 and Figure 15 shows the Training and Validation accuracy, and Training and validation loss respectively for 10 epochs for the Recurrent Neural network with Adam optimizer. It does slightly better than the previous 3 models while training. Validation accuracy does increase initially but than it increasing and remains pretty much the same just like the previous model. Also, for the loss it does decrease initially but just like the previous model it starts to increase.

Figure 16 and Figure 17 shows the Confusion Matrix and Classification report respectively for this model. Test accuracy for this model is 82.9% which is slightly better than the previous model. Precision, recall, and F1 score for the class 0 are 0.87, 0.77, 0.82 respectively. And precision, recall, and F1 score for the class 1 are 0.80, 0.89, 0.84 respectively. According to these values I can say this model performed slightly better on the class 1 when compared with the class 0 but there is not much difference. Overall, I would say the RNN model performed better when compared with all the other 3 models.

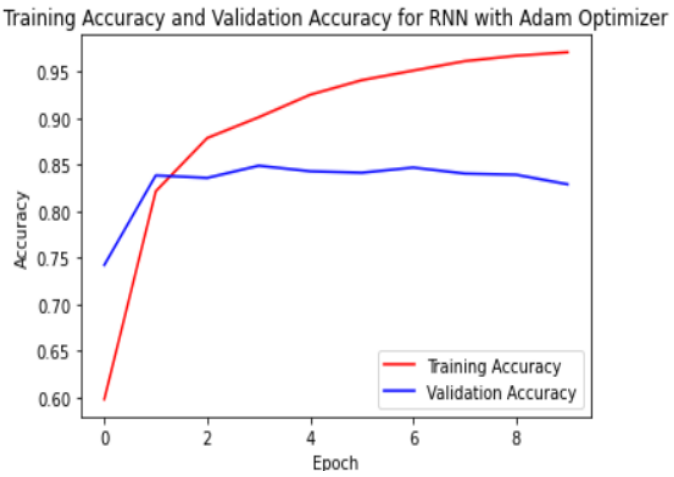

Fig. 14. Training and validation Accuracy for the RNN with Adam optimizer Model

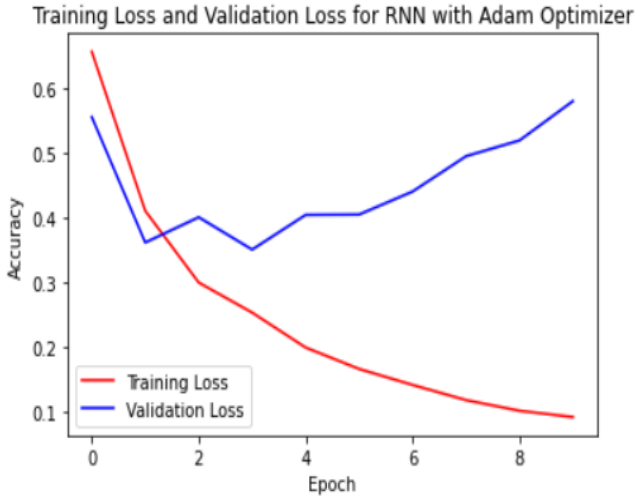

Fig. 15. Training and validation Loss for the RNN with Adam optimizer Model

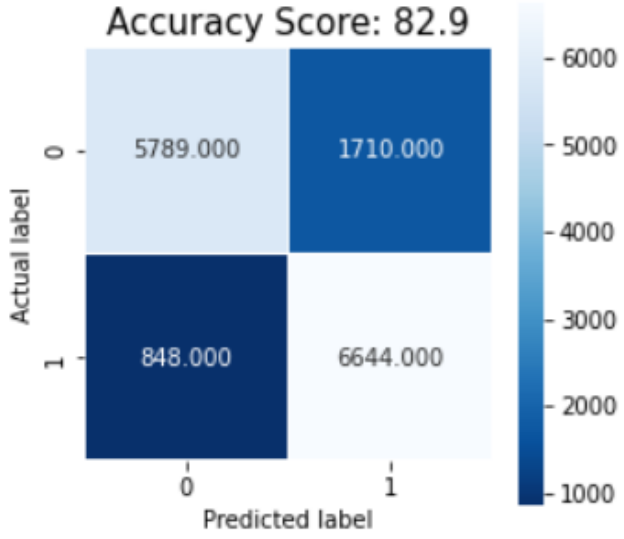

Fig. 16. Confusion Matrix for the RNN with Adam optimizer Model

	precision	recall	f1-score	support
0	0.87	0.77	0.82	7499
1	0.80	0.89	0.84	7492
accuracy			0.83	14991
macro avg	0.83	0.83	0.83	14991
weighted avg	0.83	0.83	0.83	14991

Fig. 17. Classification report for the RNN with Adam optimizer Model

One of the biggest issues I faced was getting the accuracy above 90%. Even though 82.9% is pretty good result but getting to 90 would have been even better. I tried with different architectures, different activations and learn rates but this is the best result that I got. I believe if the dataset was divided into 3 classes, positive, negative and neutral than I might have gotten a better result.

6. Conclusion

To conclude, I would say Recurrent Neural networks seems to have performed the best for the sentiment analysis task for the IMDB movie reviews and I think RNNs work best when dealing with text data because RNNs are trained to recognize patterns over time which helps it in capturing sequential data, since text is also sequential so I believe for any textual data RNNs will work the best. Even though I was not able to achieve more than 90% test accuracy, 82.9% of the test accuracy is still pretty good. One of the things I also learned was that Adam optimizer seems to have performed much better when compared with the SGD optimizer for the sentiment analysis task. For the future I would like to improve upon this model and see if I can achieve better results and also would like to work on other text analysis datasets using RNNs.

7. References

[1] Huu-Thanh Duong, Tram-Anh Nguyen-Thi, “A review: preprocessing techniques and data augmentation for sentiment analysis”, 2021. https://rdcu.be/cCChR.

[2] Rifat Rahman, Md Abdul Masud, Raonak Jahan Mimi, Mst. Nusrat Sultana Dina, “Sentiment Analysis on Adventure Movie Scripts”, 2020.

[3] Hassan Saif, Yulan He, Harith Alani, “Semantic Sentiment Analysis of Twitter”, 2012. https://rdcu.be/cCCid.

[4] Kuat Yessenov, Sasa Misailovi ˇ c, “Sentiment Analysis of Movie Review Comments”, 2009.

[5] Link to the dataset:

https://www.kaggle.com/renanmav/imdb-movie-review-dataset