NLP Sentiment Classification of Amazon Fine Food Reviews Dataset using Recurrent Neural Networks (RNN) and Long Short Term Memory Neural Networks (LSTM)

David Akinmade
ENSE 865
200440384

Abstract

With the ever reducing costs and assured investment return on online commerce, more businesses are shifting their entire operations to the internet. Key to ensuring that businesses do not fall behind in terms of maintaining their quality of service is the ability to respond quickly to the problems faced by customers, which is usually reflected in the reviews they leave behind after every transaction. This report therefore approaches the problem of text classification by making use of NLP techniques and Neural Networks (RNN and LSTM) to extract the respective overall sentiment from each row of Amazon Fine Food Review text data. The results from each model is presented and compared against each other and a baseline Logistic Regression model.

Keywords - Neural Networks, RNN, LSTM, Logistic Regression

I. INTRODUCTION

For most e-commerce companies that process a lot of orders and attend to a lot of customers on a daily basis, it is inevitable for the quality of their service delivery to fall short every now and then. However, most of these companies provide avenue for adjustment on the fly by asking for customer reviews and generally fixing the problems highlighted in problematic reviews so that no other customer needs to face similar problems. Solving this problem by reading and sorting through each customer review manually is not necessarily a bad idea especially of the business is small or doesn't process a large quantity of orders daily. But for large e-commerce companies like Amazon or Alibaba, this simple task of sorting through customer reviews scales up to become very cumbersome due to the sheer volume of reviews to be sorted through and the fact that problems need to be solved as quickly as possible within a short time-frame. Natural Language Processing (NLP)[1] is the field of Machine Learning that addresses issues of this nature where the data is textual and with the aid of machine learning models words can be mined for meaning, associations, general information or in this particular case, sentiment. This project seeks to combine the techniques of NLP with machine model learning in order to classify the data in the Amazon Fine Food Reviews dataset into positive and negative class sentiments, a Logistic Regression model is first applied in order to obtain a base level of performance and then a Recurrent Neural Network and Long Short Term Memory Neural Network are equally used and their performances compared.

This report details the individual steps carried out in this project which was carried out entirely in Python3 (using Jupyter Notebook) from the Objectives to the Methodology where the data pre-processing and transformation is discussed. After that, how each neural network modelling method and algorithm implemented used is also shown and finally, a comparative analysis is conducted on the results in order to demonstrate the abilities of each approach. In its conclusion, other related work and possible areas of model accuracy improvement are discussed.

II. RELATED WORK

A number of research works were inspirational for this project and among these many works that have been done in the field of Neural networks for solving classification problems include: the use of RNNs for hierarchical multi-class text classification [2], the use of CNNs by Zhang et al [3] for text classification with character-level features and the work by Salakhutdinov [4] that used deep learning to hierarchically classify images into their respective classes of vehicle or animal.

III. PROJECT OBJECTIVE

The goal of this project is to build a machine learning model that utilizes Neural network models to predict the appropriate sentiment label for each row of online shopping review data contained in the 'Amazon Fine Food Reviews' dataset obtained from Kaggle[5].

Expected Outcome Example:
Input:

ID	Reviews Text
0	Product arrived labeled as Jumbo Salted Peanut...
1	If you are looking for the secret ingredient i…
2	The Best Hot Sauce in the World
…	

Output:

ID	Sentiment
0	0
1	0
2	1
…	

IV. METHODOLOGY

The data used in this project was the 'Amazon Fine Food Reviews' obtained on Kaggle. The dataset is comprised of feature vectors obtained from 568454 food reviews left by 256059 different users up to October 2012. The dataset includes features like: ProductId, UserId, ProfileName, HelpfulnessNumerator, HelpfulnessDenominator, Score, Time, Summary and Text. However since the scope of this project covers only text processing and classification all of these except for Score and Text are dropped. It is also important to note that because this dataset is a standard dataset from Kaggle, the data contains no missing data or null data points.

A. *Exploratory Data Analysis*

TABLE 1: Numerical features used in the Reviews dataset

Feature name	Feature description	Min. value	Max. value	SD
HelpfulnessNumerator	Number of users who found the review helpful	0	866	7.64
HelpfulnessDenominator	Number of users who indicated whether they found the review helpful or not	0	923	8.29
Score	Rating between 1 and 5	0	1	5
Time	Timestamp for the review	939m	1.3b	48m

Table 1 above gives the statistical overview and description of all the numerical features present in the dataset, while Table 2 below gives the description of all the other non-numerical features present in the dataset.

TABLE 2: Numerical features used in the Reviews dataset

Feature name	Feature description
ProductId	Unique identifier for the product
UserId	Unqiue identifier for the user
ProfileName	Profile name of the user
Summary	Brief summary of the review
Text	Text of the review
Summary	Brief summary of the review

B. Feature Engineering and Data Selection

The data obtained did not have a "Sentiment" feature which is very important for the model training. Therefore a Sentiment feature was engineered from the Score feature through the use of a Lambda function. The lambda function is applied to every row in the Score feature such that any review that has a score of 2 and below is assigned a negative Sentiment class label of '0' and any review that has a score of 3 and above is assigned a positive Sentiment class label of '1'. The Sentiment feature is therefore engineered and chosen as the Target variable. Now that the Target feature has been engineered, the Score feature is also dropped and only the Text and Sentiment features are chosen as the input and output variables respectively

TABLE 3: Raw features before Feature Engineering and Selection

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	Summary	Text
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	1	5	1303862400	Good Quality Dog Food	I have bought several of the Vitality canned d...
1	2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	0	1	1346976000	Not as Advertised	Product arrived labeled as Jumbo Salted Peanut...
2	3	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	1	1	4	1219017600	"Delight" says it all	This is a confection that has been around a fe...
3	4	B000UA0QIQ	A395BORC6FGVXV	Karl	3	3	2	1307923200	Cough Medicine	If you are looking for the secret ingredient i...
4	5	B006K2ZZ7K	A1UQRSCLF8GW1T	Michael D. Bigham "M. Wassir"	0	0	5	1350777600	Great taffy	Great taffy at a great price. There was a wid...

TABLE 4: Data after Feature Engineering and Data Selection

	Text	Sentiment
0	I have bought several of the Vitality canned d...	1
1	Product arrived labeled as Jumbo Salted Peanut...	0
2	This is a confection that has been around a fe...	1
3	If you are looking for the secret ingredient i...	0
4	Great taffy at a great price. There was a wid...	1
5	I got a wild hair for taffy and ordered this f...	1

After the encoding, the target feature 'Revenue' was then removed from the dataset and the remaining features were transformed by using StandardScaler from Scikit-learn to scale them such that their mean becomes zero and the standard deviation 1. This is necessary so as to prevent data points from features with high numerical values from unfairly skewing the predictions in any particular way, therefore leading to more accurate predictions by the Ensemble models.

C. Text Tokenization

The following step was the tokenization of the data in the Text feature into a numerical vector of numbers representing the texts in each row of data. This was achieved in two steps: First the texts from each row in the dataframe was extracted and appended to a list, and then the Tokenizer function from the Natural Language Tool Kit (NLTK) is fitted on this list of texts in order to generate a sequence list for each row which is then converted to an np array. After that the sequences are padded with zero values to ensure they are all of dimension 100.

```
In [25]: 1 print(sequences)

[list([9, 11, 51, 8, 39, 412, 6, 151, 17, 217, 15, 19, 17, 5, 182, 6, 41, 87, 343, 5, 133, 63, 3, 228, 2, 277, 11, 1, 346, 7])
 list([2, 26, 217, 820, 20, 33, 7, 13, 152, 6, 274, 30, 71, 35, 15, 2, 279, 44, 5, 827, 3, 727, 58, 1, 8, 276, 2, 60, 6, 3, 13
2, 2, 141, 13, 208, 22, 343, 5, 34, 47, 2, 630, 57, 409])
 list([9, 8, 1, 85, 812, 5, 2, 76, 6, 11, 41, 13, 567, 45, 3, 89, 12, 510, 567, 429, 39, 59, 31, 270, 59, 948, 3, 59, 473, 9, 1
64, 50, 559, 59, 594, 11, 51, 31, 470, 457, 137, 54, 6, 73, 92, 4, 37, 800, 26, 1, 196, 3, 36, 19, 6, 6, 3, 130, 1, 121, 6, 55,
24, 189, 303, 460, 6, 31, 19, 55, 17, 5, 4, 144, 34, 13])
 ...
 list([2, 53, 9, 39, 24, 104, 315, 89, 515, 417, 150, 896, 5, 238, 31, 662, 4, 193, 302, 641, 4, 179, 7, 42, 4, 402, 391, 389,
65, 289, 398, 38, 569, 271, 387, 4])
 list([2, 330, 27, 22, 78, 34, 25, 29, 41, 7, 28, 3, 22, 322, 34, 23])
 list([2, 21, 5, 17, 333, 27, 12, 1, 2, 116, 5, 102, 27, 24, 4, 235, 47, 2, 12, 195, 27, 20, 5, 434, 2, 996, 5, 17, 1, 349, 32
8, 57, 19, 17, 5, 220, 3, 76, 4, 5, 60, 1, 349, 5, 57, 25, 331, 178, 380, 18, 23, 223, 23, 331, 26, 96, 33, 2, 416, 63, 2, 185,
279, 26, 4, 21, 2, 21, 34, 383])]
```

Figure 1: Text feature after been tokenized

```
In [26]: 1 print(x_train)

[[  0   0   0 ...   1 346   7]
 [  0   0   0 ... 630  57 409]
 [  0   0   0 ... 144  34  13]
 ...
 [  0   0   0 ...  84 112 303]
 [  0   0   0 ... 424   7 859]
 [  0   0   0 ... 776   7  68]]
```

Figure 2: Tokenized Text feature after padding

V. Modelling

A. Data Splitting and Balancing

After the data was successfully pre-processed and tokenized, it was fed into the models. But before doing that two key steps were carried out:

1. Data splitting

The data is split 70% train data and 30% using the Train-Test-Split function. The stratify parameter is also set on the target feature to ensure that the number of number of data points from each class in the target feature is proportionally split into the train and test sets. 70% of the data was for training used so as to maximize the prediction accuracy of the models especially on the data points that have a class Sentiment value of 0 as the data is imbalanced, having more data points that have a Sentiment class label of 1.

2. Data balancing

To finally address the issue of data imbalance, the data points in the training set where Revenue = 1 are over-sampled using the SMOTE function from the Imbalanced-Learn python library to match the number of data points where Sentiment = 0. This raises the total number of data points from 390000 to 667762, where the new dataset has 333881 data samples with Sentiment class of 1 and also 333881 data samples with Sentiment class of 0, therefore balancing the total number of positive and negative target class labels.

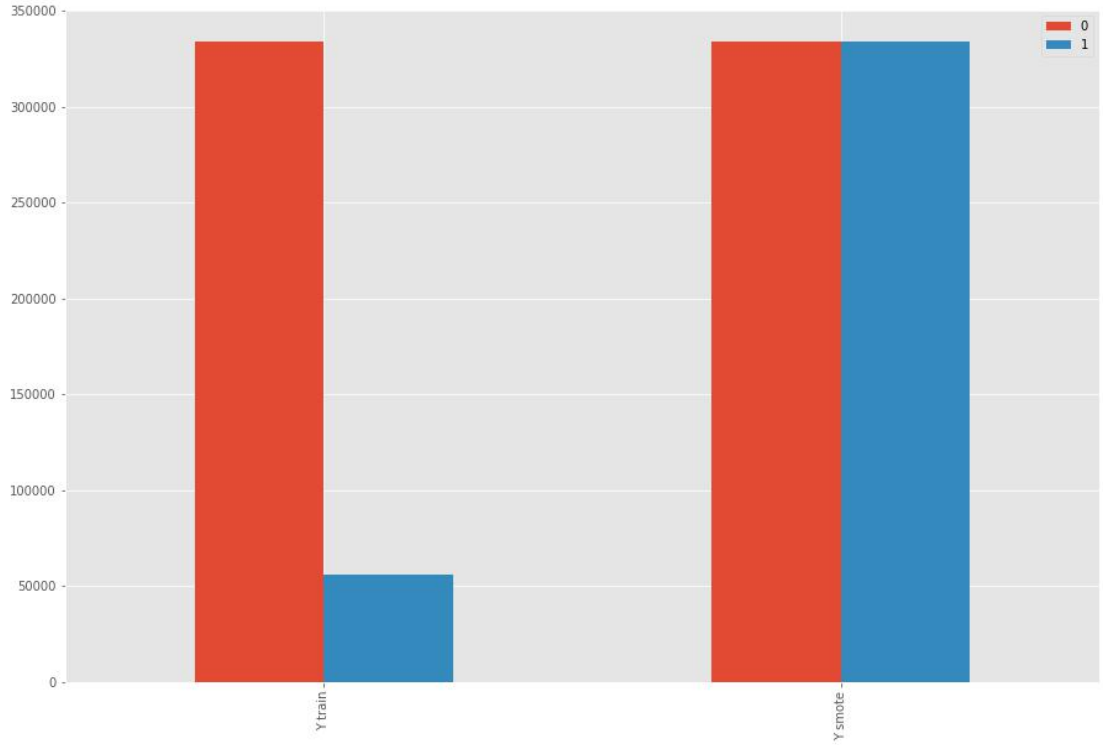

Figure 3: Data samples of the Target variable before and after SMOTE balancing

B. Base Model

To set the tone for the entire modelling exercise, a base level of performance has to be set in order to judge if there is a progression in the prediction capability of each successive model. For this project, the base model that was used was a simple Logistic Regression model[6]. This was chosen due to its simplicity as Logistic Regression models work by applying a sigmoid function to a loss function obtained by summing up the logarithm of the likelihood function.

A lot of emphasis would be placed on F1-scores because F1-score is the parameter that shows how well our precision and recall scores balance each other.

Below is the performance of the base model on the test data

TABLE 5: Base Model Performance

Target Class Value	Precision	Recall	F1-score	Support
0	0.14	0.51	0.22	25918
1	0.85	0.48	0.62	152536
Macro Avg	0.50	0.49	0.42	178454
Accuracy				0.49

From the performance of Logistic Regression, we obtained an average F1- score of 0.42 and an overall model accuracy of 0.49 which is okay for a benchmark performance as it is about the same level of accuracy as a random chance prediction. However, the F1-score for 0 label data points in the target class is low at 0.22. Therefore we expect future models to perform better at predicting the negative target class labels by having a much better F1-score.

D. *Neural Network Models*

Two neural networks were used and they are:

1. Recurrent Neural Networks (RNN)

RNNs[7] are a generalization of feed-forward neural networks built specifically to handle sequential data e.g Text. This means that instead of a traditional neural networks where all input and output weights are not dependent on each other, we have a situation where every input of data has the same function performed on it recurrently and the output of the current input is reliant on the one from the last computation. RNNs therefore have a sense of memory, as they are able to collate information about what has been calculated so far for each element in a sequence, and this information is stored

as its hidden state which is the main component of an RNN. However, it is important to note that RNNs are limited in their ability to capture long-term dependencies and are more likely to make predictions based on the most recent elements in a sequence. In the figure below, x_1, x_2, x_3, …, x_t represent the input words from the text, o_1, o_2, o_3, …, o_t represent the predicted next words and h_0, h_1, h_2, h_3, …, h_t hold the information for the previous input words.

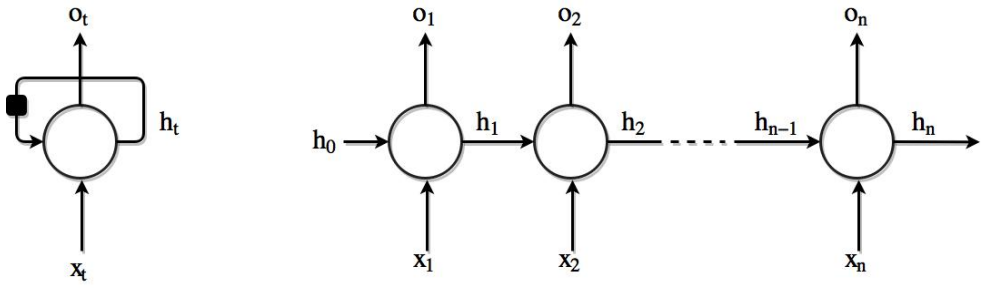

Figure 4: Basic structure of an unrolled RNN[8]

2. Long Short Term Memory (LSTM) Neural Networks

LSTMs are basically just an enhanced version of RNNs as they are engineered to remember past data in the hidden state much easier therefore resolving the vanishing gradient problem of RNNs. LSTMs are trained using the process of backward propagation. An LSTM model has three main components:

Input gate: which identifies what value of input should be used to modify the memory

Forget gate: which discovers what details are to be removed from memory of the block

Output gate: which computes the output value based on the input value and the memory of the block

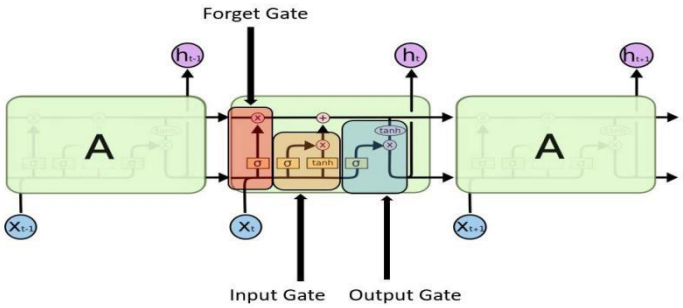

Figure 5: Basic structure of an unrolled RNN[9]

E. Model Implementation

The first Neural network model called ‘model_rnn’ is structures as follows:

Layer 1:

The first layer is an Embedding layer (which is typically used when dealing with text data) initialized with the maximum length of the input data columns (100) and its vocabulary size (1000) it is then followed by a Dropout function that randomly excludes 30% of the data going into the next layer. This is done so as to reduce overfitting in the overall neural network model. Finally a BatchNormalization function is applied to normalize the data on its way out from the embedding layer.

Layer 2:

The second layer is where the RNN is situated and it is initialized to receive and process the 32 outputs from the Embedding layer. Here also a Dropout function and BatchNormalization function are applied.

Layer 3:

The final layer is the dense layer where the final output/prediction is made. It is initialized to return a single value as output and it is fitted with a sigmoid function to collapse the binary list of probabilities into just 1. After that the model is compiled using RMSprop as the gradient optimization technique and Binary Cross-Entropy as the loss function

```
model_rnn = Sequential()
model_rnn.add(Embedding(1000, 32, input_length=maxlen))
model_rnn.add(Dropout(rate=0.3))
model_rnn.add(BatchNormalization())
model_rnn.add(SimpleRNN(32))
model_rnn.add(Dropout(rate=0.3))
model_rnn.add(BatchNormalization())
model_rnn.add(Dense(1, activation='sigmoid'))
model_rnn.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['acc'])
model_rnn.summary()
history_rnn = model_rnn.fit(x_train_smote, y_smote, epochs=8, batch_size=256, validation_split=0.2)
```

Figure 6: Architecture of implemented RNN model

The second model implemented called 'model_lstm', follows the exact same architecture as the first one only that in the second layer the RNN network is replaced with an LSTM network.

```
model_lstm = Sequential()
model_lstm.add(Embedding(1000, 32, input_length=maxlen))
model_lstm.add(Dropout(rate=0.3))
model_lstm.add(BatchNormalization())
model_lstm.add(LSTM(32))
model_lstm.add(Dropout(rate=0.3))
model_lstm.add(BatchNormalization())
model_lstm.add(Dense(1, activation='sigmoid'))
model_lstm.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['acc'])
model_lstm.summary()
history_rnn = model_lstm.fit(x_train_smote, y_smote, epochs=8, batch_size=256, validation_split=0.2)
```

Figure 7: Architecture of implemented LSTM model

VI. RESULTS

A. Performance Metrics

The following are the performance metrics used in evaluating and comparing the performances of each model:

1. Accuracy: total number of correct predictions / total number of predictions
2. Precision: true positive/ (true positive + false positive)
3. Recall: true positive/ (true positive + false negative)
4. Support: This refers to the number of data samples used from each class
5. F1-score: This refers to the ratio of the product of Precision and Recall to their sum multiplied by two. This can be expressed mathematically as:

$$F_1 = \frac{2(\text{Precision x Recall})}{(\text{Precision} + \text{Recall})}$$

The F1-score is very important in this project, especially the F1-score on negative class data samples since it is very important for online companies to know how to predict which reviews have negative sentiments as those are the orders with problems that need fixing.

B. Results

TABLE 6: RNN model Performance

Target Class Value	Precision	Recall	F1-score	Support
0	0.70	0.65	0.67	25918
1	0.94	0.95	0.95	152536
Macro Avg	0.82	0.80	0.81	178454
Accuracy				0.91

TABLE 7: LSTM model Performance

Target Class Value	Precision	Recall	F1-score	Support
0	0.78	0.70	0.74	25918
1	0.95	0.97	0.96	152536
Macro Avg	0.87	0.84	0.85	178454
Accuracy				0.93

TABLE 8: Overall Model Performance Comparison on Test Data

Model	Avg Precision	Avg Recall	Avg F1-score	Accuracy
Logistic Regression	0.50	0.49	0.42	0.49
RNN	0.82	0.80	0.81	0.91
LSTM	**0.87**	**0.84**	**0.85**	**0.93**

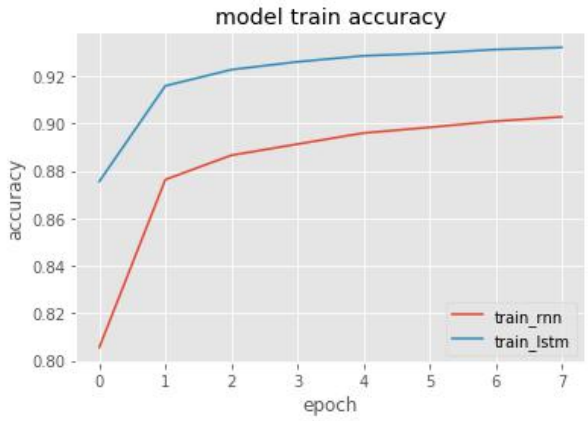

Figure 8: Comparison of RNN and LSTM model accuracy on training data

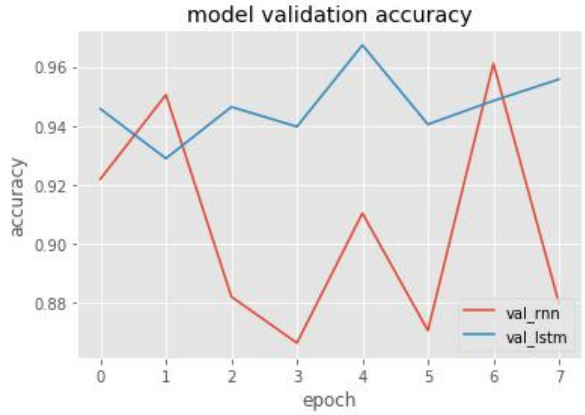

Figure 9: Comparison of RNN and LSTM model accuracy on validation data

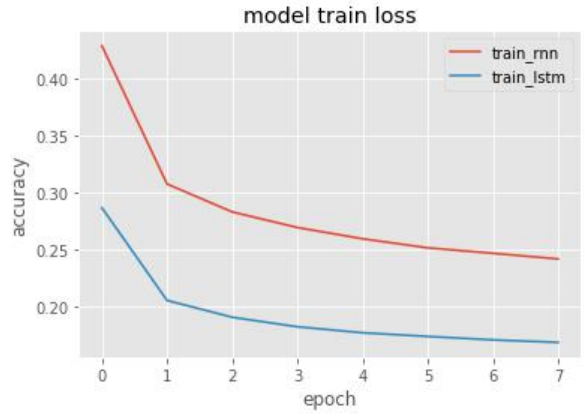

Figure 10: Comparison of RNN and LSTM model loss on training data

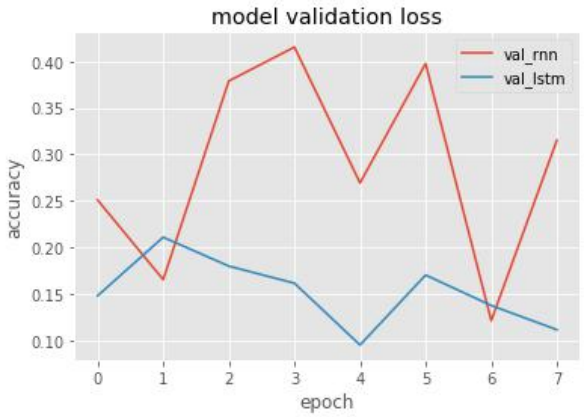

Figure 11: Comparison of RNN and LSTM model loss on validation data

The results at the end of modelling show the following:

1. Both Neural network models perform vastly better than the chosen base model of Logistic Regression, which is an indication of the success of the project
2. The RNN model performs very well with an overall accuracy of 0.91 and has 0.82, 0.80 and 0.81 as its respective average precision, recall and f1 score. It also performs slightly above average in the prediction of negative class labels as it has an f1 score of 0.67.
3. LSTM as was predicted performs better than RNN with an overall accuracy of 0.93 and has 0.87, 0.84 and 0.85 as its respective average precision, recall and f1 score. However where it truly shows its superiority is in its ability to predict negative target class samples better than RNN as it had an f1 score of 0.74 in that aspect.

VII. LIMITS and EXTENSION

The scope of this work covered only the application of RNNs and LSTM neural networks to solve classification problems. However it is important to note that other neural network models such Transformers and word tokenization methods like BERT could be applied in order to obtain even more precise models with higher levels of accuracy and performance on the prediction of negative class target data.

VIII. CONCLUSION

In conclusion, this project has been able to show that neural networks can be successfully applied to NLP binary classification machine learning problems. Also the supremacy of LSTMs over RNNs was shown as LSTMs were able to get even higher accuracy and F1 scores owing to its far superior memory short term memory system.

IX. REFERENCES

[1] Kamal Nigam, et al (1999). Text Classification from Labeled and Unlabeled Documents using EM. Machine Learning.
[2] Ruseti, S., Dascalu, M., Johnson, A., Balyan, R., Kopp, K., McNamara, D., Trausan-Matu, S. (2018). Predicting Question Quality Using Recurrent Neural Networks. 10.1007/978-3-319-93843-1_36.
[3] Zhang, X.,Zhao, J., and Y. LeCun, Y. (2015). Character-level convolutional networks for text classification in Advances in neural information processing systems pp. 649–657.
[4] R. Salakhutdinov, R., J. B. Tenenbaum, J.R., and A. Torralba, A.(2013). Learning with hierarchical-deep models. IEEE transactions on pattern analysis and machine intelligence, vol. 35, no. 8, pp. 1958–1971.
[5] https://www.kaggle.com/snap/amazon-fine-food-reviews
[6] Cox, D.R. (1972). Regression Models and Life Tables. Journal of the Royal Statistical Society, Series B, 34, 187- 202.
[7] Hopfield, J.J (1982). Neural networks and physical systems with emergent collective computational abilities. Proceedings of the National Academy of Sciences 2554-2558.

[8] https://blog.paperspace.com/recurrent-neural-networks-part-1-2/
[9] https://towardsdatascience.com/understanding-rnn-and-lstm-f7cdf6dfc14e