Aspect-based Sentiment Analysis Algorithm based on Syntactic Parsing

By Ankita Choudhury

Abstract

Classification of text to predict its sentiment is a part of Natural Language Processing (NLP). NLP is a branch of computer science which helps the computer to read, understand and interpret human speech. Dependency parsing was used to extract relevant information from a review in order to predict the sentiment of a given aspect term. Different machine learning models such as Naïve Bayes, Logistic Regression, Support Vector Classifier and Neural Networks were used to make predictions. A maximum accuracy score of 0.74 on the test dataset was achieved using Bidirectional LSTM model.

Introduction

A restaurant review is a type of evaluation of a restaurant. It can be brief or in-depth. Restaurant reviews are provided by people who have already dined in at a restaurant. Recognizing the sentiment of a restaurant review is a method of determining whether people liked or disliked a particular restaurant. Reviews help to provide some information about the ambience, service and food of a restaurant.

Classification of aspect terms in restaurant reviews as positive, negative or neutral is a part of Natural Language Processing (NLP). NLP helps computers read, understand and interpret human language. It is being widely used for sentiment analysis, language translation, text extraction, chatbots etc. Many different types of algorithms are used to perform text classification. With the increasing number of online platforms containing comments, reviews etc, the task of assigning a label to texts has high practical significance [1]. It can be used to filter out toxic information or to simply understand the sentiment of text. Logistic Regression, Naive Bayes, Support Vector classifier etc. are some of the methods of performing text classification.

Dataset

The dataset used for this project is a restaurant review dataset. The dataset is a collection of reviews of restaurants. The training data is a xml file named "Restaurants_Train.xml'" and the testing data is "Restaurants_Test_Gold.xml". Both files contain sentence id, text, aspect term, aspect term polarity, aspect category and aspect category polarity. The text contains the review text. The aspect term is the term for which the sentiment will be analysed. There are 3041 reviews in the training set and 800 the testing set. There are 2021 reviews containing at least one aspect term in the training set and similarly 606 in testing set. There are three possible values for polarity or sentiment of a term, namely, positive, negative and neutral. The distribution of classes is shown in Figure 1 and 2. Most of the polarity values are positive in both training and testing sets. This problem is a multi-class classification problem as there are more than two classes.

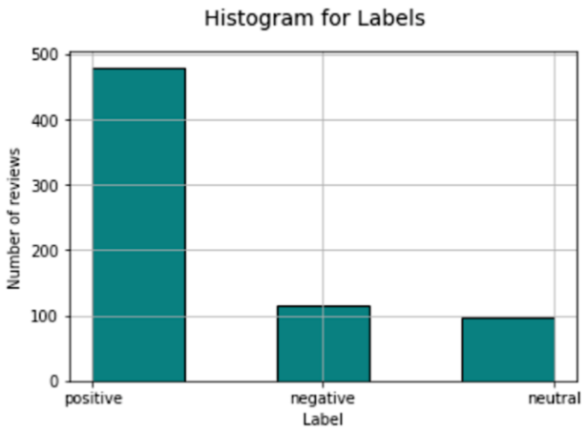


Figure 1: Distribution of Label in the Training dataset

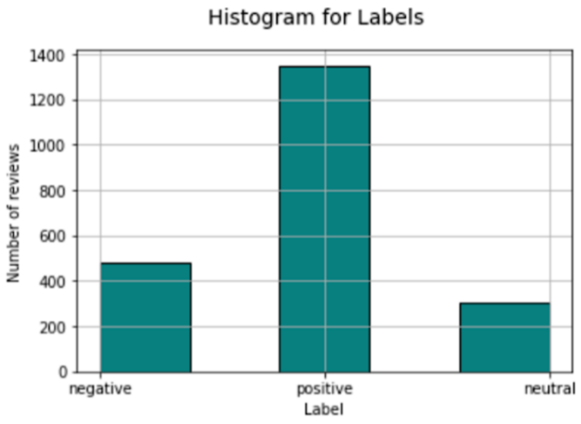


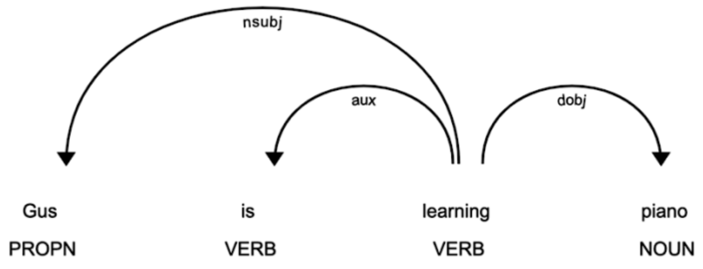
Figure 2: Distribution of Label in the Test dataset

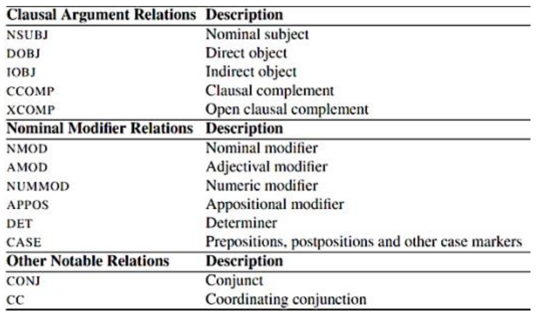
Data Loading

A Jupyter notebook with Python kernel was used to perform all the tests on Google Collab. Natural Language Toolkit (NLTK) was the primary library used for processing text. NLTK is one of the most popular libraries used for Natural Language Processing. NLTK was used to perform tokenization, lemmatization, stop word removal etc. Sklearn, Pandas and NumPy library were also frequently used for data loading, manipulation and transformation. Matplotlib and seaborn library was used to plot visualisations. The xml file was loaded with the help of Element Tree Library in Python. All the review text, aspect term and polarity for both the training and testing data were extracted using processXML() function defined in the code.

Syntactic Dependency Parsing Method

The process of extracting a sentence's dependency parse to represent its grammatical structure is known as dependency parsing. Dependency parsing can be used to create an accurate dependency tree and tag each word with the appropriate dependency tag. A dependency tree is a directed graph in which one word is connected to another by an arrow pointing from the head word to the dependent words. Spacy library offers a fast and accurate dependency parser. The parser can also detect sentence boundaries and iterate over base noun phrases.





Clausal Argument Relations	Description
NSUBJ	Nominal subject
DOBJ	Direct object
IOBJ	Indirect object
CCOMP	Clausal complement
XCOMP	Open clausal complement
Nominal Modifier Relations	**Description**
NMOD	Nominal modifier
AMOD	Adjectival modifier
NUMMOD	Numeric modifier
APPOS	Appositional modifier
DET	Determiner
CASE	Prepositions, postpositions and other case markers
Other Notable Relations	**Description**
CONJ	Conjunct
CC	Coordinating conjunction

Figure 3: Dependency parsing example

A system was designed which extracts the head.children and token.children for a given token. Here token refers to an aspect term in a review. It then combines all the words in the head.children and token.children to form a single text sentence. These two terms were combined to extract all the relevant information required to predict the sentiment for a token or aspect. The function getDictData() returns a dictionary with the combined text for all aspect terms as key and its polarity is the value of the key. The dictionary is then converted to a data frame with two columns, text and sentiment.

Pre-Processing Text

The rows with sentiment as conflict were removed from both the training and testing data frame. Some pre-processing was done as described below:

Tokenisation and Stop word removal

A function to perform text cleaning was implemented. The function 'clean_corpus' first converts all input text to lower case. This is done as some models are case sensitive. Converting all text to lower case might be counterproductive for some words like 'US' as they lose semantic their meaning. Overall, it is still a good method for pre-processing text. Then the text data was tokenized using word_tokenize function from NLTK. Stop words are the commonly used words in the English language. They carry little meaning and have low importance for predictions. The stop words were removed from the tokenized text. The stop words that were removed were predefined stop words in the English language by NLTK library. After that, all the punctuations were also removed from the tokens. The text which was numerical was also removed.

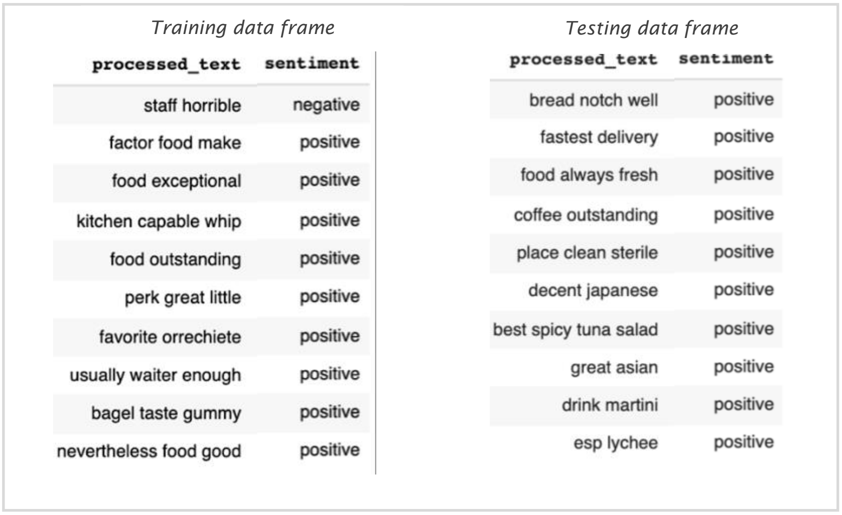
Lemmatization

Lemmatization is the process of finding the lemma of a word based on its meaning. Lemmatization aims to remove inflectional endings. It aids in returning a word's base form. NLTK provides a lemmatizer based on

WordNet built in morphy function. If any word is not found in Wordnet, it returns the word unchanged. Lemmatization is a good method to preprocess text as it always produces existing words in the English language.

Drop duplicates and text with only one word

After performing lemmatization, any duplicate rows in the text column were dropped for both training and testing data. Sentences with only one word were also removed as they did not carry any information about the sentiment of an aspect.



Training data frame

processed_text	sentiment
staff horrible	negative
factor food make	positive
food exceptional	positive
kitchen capable whip	positive
food outstanding	positive
perk great little	positive
favorite orrechiete	positive
usually waiter enough	positive
bagel taste gummy	positive
nevertheless food good	positive

Testing data frame

processed_text	sentiment
bread notch well	positive
fastest delivery	positive
food always fresh	positive
coffee outstanding	positive
place clean sterile	positive
decent japanese	positive
best spicy tuna salad	positive
great asian	positive
drink martini	positive
esp lychee	positive

Figure 3: Some texts after pre-processing

TF-IDF

TF-IDF was used next to transform the text column in the dataframe to a vector to be able to fit to a model.TF-IDF stands for Term Frequency - Inverse Document Frequency. TF IDF is used to quantify words in a document by assigning a weight to each word which represents the importance of the word in the document. The formula for calculating TF IDF is

TF-IDF = Term Frequency (TF) * Inverse Document Frequency (IDF)

Term Frequency (TF)

This metric counts the number of times a word appears in a document. Certain words like ‘the’, ‘and’, ‘is’ etc. appear many times in a document. Thus, they have a high frequency.

Term Frequency of a document = Count / Total number of words

To calculate term frequency for a document, the frequency of a word in a document is divided by the total number of words in the document. This is a way to normalize the frequency for a document. Documents containing many words would not have greater importance than a document containing a smaller number of words.

Inverse Document Frequency (IDF)

A document frequency of a word is the number of occurrences of it in different documents. It is the number of documents in which a word appears. The number of times a word appears in a single document is not important.

Inverse Document Frequency = Count of Documents / Document Frequency

Inverse Document Frequency of a word is calculated by dividing the total number of documents by the document frequency of a word. Since common words like stop words appear a lot of times in a large number of documents, their IDF would still be small number.

Evaluation Metric

Accuracy was the primary metric used to evaluate the model performance. The accuracy score is computed as:

$$Accuracy = \frac{True\ Positive + True\ Negative}{True\ Positive + True\ Negative + False\ Negative + False\ Positive}$$

True positive is calculated by summing all the instances where the actual and predicted label both are positive. True negative is calculated by summing all the instances where the actual and predicted label both are negative. False positive is calculated by summing all the instances where the actual label is negative, but the predicted label is positive. False negative is calculated by summing all the instances where the actual label is positive, but the predicted label is negative.

Pandas and seaborn library were used to plot a confusion matrix. In a confusion matrix, the left diagonal boxes represent the number of correct classified samples and the right diagonal boxes represent the number of incorrect classified samples as illustrated in Figure 4.

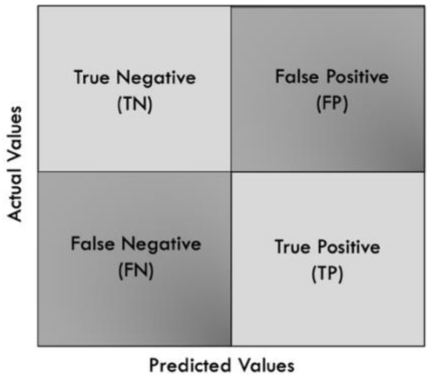


Figure 4: Confusion Matrix and its Constituents

Data Modelling

For this project, various different types of models were implemented. Initially, Logistic Regression, Multinomial Naïve Bayes and Support Vector Classifier was used to model the data transformed by TF-IDF. Grid search was used to tune the hyperparameters in these models. Cross validation sets were developed using Stratified K Fold. Stratified K Fold ensures that each fold of the dataset has the same proportion of observations with a given label. The Different parameters fit to these models and the resulting accuracy on cross validated sets is shown in the Table 1.

Logistic Regression			
Solver	**C**	**Tolerance**	**Mean Accuracy**
Lbfgs, newton-cg, sag, saga	0.01, 0.1	0.0001	0.637
Lbfgs, newton-cg, sag	0.5	0.0001	0.659
saga	0.5	0.0001	0.660
Lbfgs, newton-cg, sag, saga	1.0	0.0001	0.685
lbfgs, newton-cg, sag	1.5	0.0001	0.693
saga	1.5	0.0001	0.692
Support Vector Classifier			
Kernel	**C**	**Gamma**	**Mean Accuracy**
Rbf, poly, sigmoid	0.1,0.5, 1.0,5.0	0.0001	0.637
rbf	5.0	0.1	0.680
sigmoid	5.0	0.1	0.663
Rbf, poly, sigmoid	10.0	0.0001	0.637
rbf	10.0	0.1	0.690
sigmoid	10.0	0.1	0.688
Multinomial Naïve Bayes			
Value of Alpha	**Mean Accuracy**		
0.001	0.656		
0.01	0.656		
0.1	0.671		
0.5	0.681		
1.0	0.657		
1.2	0.653		

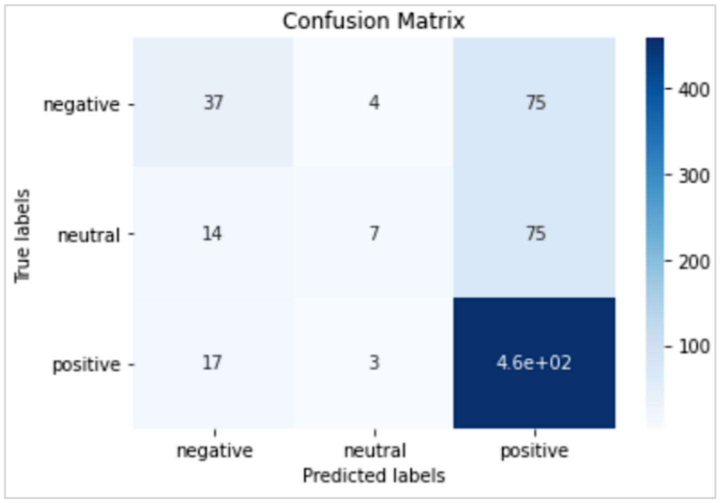
Table 1: Different parameters for different models with the accuracy achieved

The best accuracy on the validation set by model is shown in the table below.

Model	**Best accuracy on validation set**
Logistic Regression	0.693
Support Vector Classifier	0.690
Multinomial Naïve Bayes	0.681

Table 2: Best accuracy by model

It can be observed from Table 2 that Logistic Regression model has the highest accuracy of 69.3% on the cross-validation data. This is achieved by setting parameter C to 1.5 and tolerance value to 0.0001. All three solvers lbfgs, newton-cg, sag have the same resulting accuracy. This model was then used to predict the sentiment on the test data. The confusion matrix plot is shown in figure 5.



Test set
Accuracy: 0.7293

Figure 5: Confusion Matrix and accuracy for Logistic Regression on test set

Logistic Regression model achieves an accuracy of 73.9% for the test data set. It can be observed that the positive class is has the highest true positive rate. However, the neutral and negative class samples are mostly misclassified as positive.

Neural Networks

Two different types of neural network architectures were implemented to predict the sentiment. First a convolutional network with two convolutional layers and one dense layer was defined. A Bidirectional LSTM neural network was also implemented. Pre trained embeddings were used to transform the processed text to vector. The pre trained embedding used for this project is the Glove 100-dimensional word embedding. This embedding contained 400000 words and their vector representation. Glove is an unsupervised learning algorithm which can maps words together in a space using semantic similarity. It is open sourced and can be downloaded from "http://nlp.stanford.edu/data/glove.6B.zip".

Convolutional Neural Network (CNN)

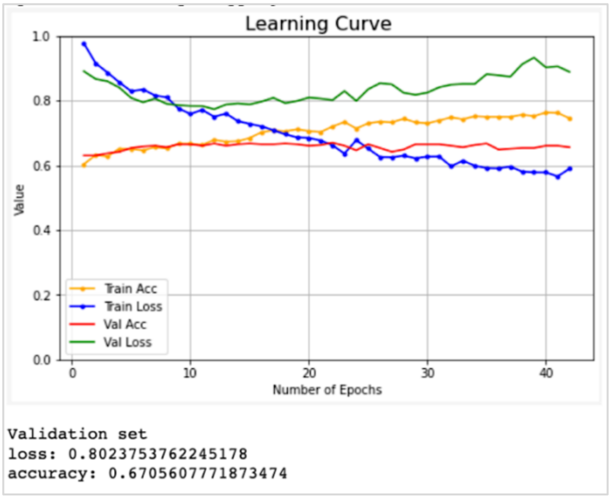
Convolutional neural networks perform well when there is a local relationship present in the data. Therefore, it usually used to model image data. However, it can also be used to solve text classification problems. A convolution in the context of a convolutional neural network is a linear operation that involves the dot product of a set of weights called filters with the input. The CNN network designed in this project has an embedding layer as the first layer. The embedding layer is followed by a dropout layer with 0.4 dropout rate. The next layer is a convolutional layer with 32 filters and 3 strides. Its activation function is Relu. This layer is followed by a 1-d max pooling layer. The next layer is another convolutional layer with 64 filters and 3 strides, followed by a global max pooling layer. Then there is a dense layer with 6 filters, which is followed by second dropout layer with 0.4 dropout rate. The final output layer is next, which has 3 channels and softmax as activation function. The model is compiled with categorical crossentropy loss and Adam optimizer. The model is fit with 100 epochs and early stopping criterion which monitors validation accuracy. The batch size was set to 20.

Bidirectional Long-Term Short-Term Memory Network (LSTM)

An LSTM layer consists of a set of recurrently connected blocks, known as memory blocks. Bidirectional LSTM is an advanced version of LSTM. One limitation of conventional recurrent neural network (RNN) is that they can only use previous context. Bidirectional RNN overcomes this limitation by processing data in both directions using two separate hidden layers that are then fed forwards to the same output layer. When Bidirectional RNN and LSTMs are combined, bidirectional LSTM is created. Bidirectional LSTM can access long-range context in both input directions [2].

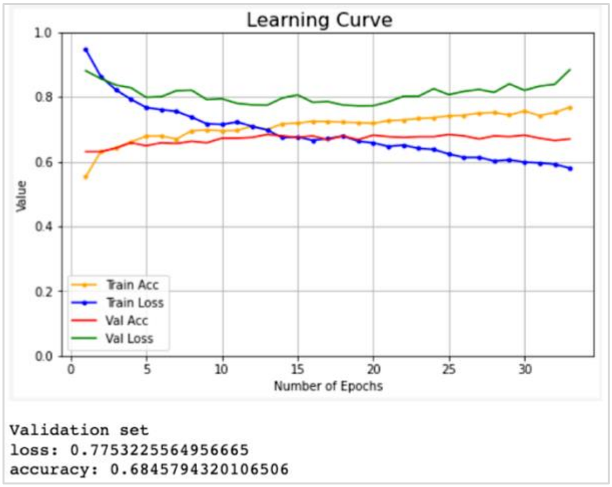
The LSTM network designed also has the embedding layer as the first layer. It is followed by a bidirectional LSTM layer with 20 units and dropout rate of 0.5. The next layer is a dense layer with 10 filters and relu activation. It is followed by the final dense layer with 3 output channels and softmax activation function. The model is compiled with categorical crossentropy loss and RMSprop optimizer. The model is fit with 100 epochs and early stopping criterion which monitors validation accuracy. The batch size was set to 20.

After fitting training data and setting validation set as validation data, the following results were obtained.



```
Validation set
loss: 0.8023753762245178
accuracy: 0.6705607771873474
```

Figure 6: Learning curve ,loss and accuracy for CNN Model on validation set



```
Validation set
loss: 0.7753225564956665
accuracy: 0.6845794320106506
```

Figure 7: Learning curve, loss and accuracy for BiLSTM Model on validation set

It can be observed that the LSTM Model has lower loss and higher accuracy than the CNN Model. Therefore, I will try to optimise the the LSTM Model model further by some hyperparameter tuning.

<u>LSTM Optimisation</u>

Batch size and Epoch

First, different values of batch size and epoch were used to fit the model using Keras Classifier and Grid Search CV. It was observed that the model with a batch size of 128 and number of epochs as 50 had the highest accuracy on validation set. These values were set in the LSTM model and were used to do further optimisation.

Optimiser

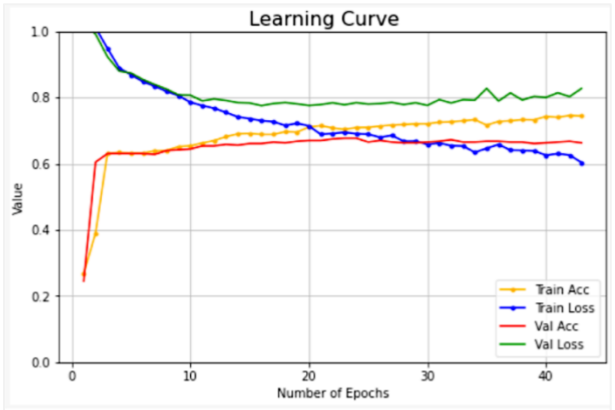
Different optimisers like 'SGD', 'RMSprop', 'Adagrad', 'Adadelta', 'Adam', 'Adamax', 'Nadam' were used to compile the model. It was found that the optimizer Adam provided the highest accuracy of 0.693.

Number of units in LSTM layer and Number of filters in first Dense layer

Different number of units were used to train and fit the LSTM Model. Specifically, number of units were set to 2, 5, 10, 15, 20, 25 and 30, and the dense layer filters were 10. It was found that number of units set to 20 provided the highest accuracy. Next, different number of dense filters were used to fit the model with LSTM layer unit set to 20. Filters were set to 6, 9,10,12 and 16. It was observed that setting the dense layer with 10 filters provided the highest accuracy.

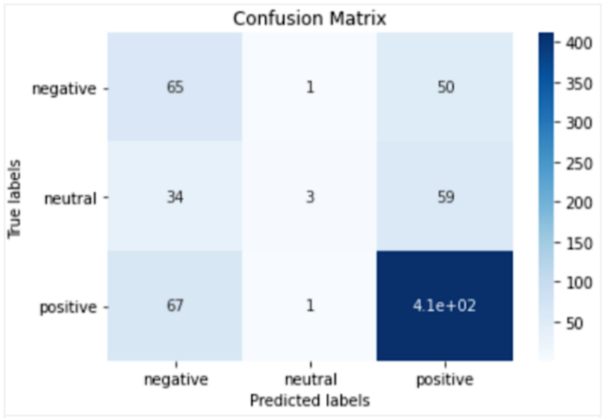
Final Model

After tuning these hyperparameters, the best performing parameters i.e., batch size of 128, optimizer as Adam, units in LSTM layer as 20 and 10 filters in dense layer, were used to fit the BiLSTM model. Number of epochs was set to 60 with and early stopping criterion monitoring validation accuracy. After fitting the training data, the validation set achieved an accuracy of 67.77% and the test set achieved an accuracy of 74.24%.



```
Validation set
loss: 0.7787066102027893
accuracy: 0.677570104598999
```

Figure 8: Learning curve, loss and accuracy for final BiLSTM Model on validation set



```
Test set
loss: 0.6610619425773621
accuracy: 0.7424023151397705
```

Figure 9: Confusion Matrix, loss and accuracy for final BiLSTM model on test set

The model showed better performance on the test data. A learning curve for the training and validation loss and accuracy for the final model is displayed in figure **x**. After epoch 30, the validation loss starts to increase. The validation accuracy is also almost constant at this point. A confusion matrix plot for the test predictions is shown in figure **x**. It can be observed that the positive class is predicted most accurately (86% true positive rate). The neutral class have very low true positive rate (3%). It is mostly misclassified as positive (61% of actual neutral text). The negative class has a better true positive rate (56%) but almost half of its samples are classified as positive.

Conclusion

An Aspect-based Sentiment Analysis Algorithm was build using Dependency Parsing and machine learning models. Accuracy score was the primary metric used for evaluating the model. A maximum accuracy score of 74.24% was achieved using the Bidirectional LSTM model on the test set. There is a scope of improving the model's performance further. The final is not able to classify the classes neutral and negative accurately. A larger training data and further optimisation of model can be helpful for predicting these classes correctly. Other neural network model architectures can be experimented with, in order to improve test performance.

References

[1] Xu, S 2018, 'Bayesian Naïve Bayes classifiers to text classification', Journal of Information Science, vol. 44, no. 1, pp. 48–59.

[2] Graves, A, Mohamed, A & Hinton, G 2013, 'Speech recognition with deep recurrent neural networks', in 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, IEEE, pp. 6645–6649.