Summer of Science Final Report

IIT Bombay

Natural Language Processing: Sentiment analysis of twitter data using LSTM Recurrent Neural Network

Submitted by
Aazad Patle
Roll No 193310001
M. Tech, Geoinformatics, CSRE
IIT Bombay

Mentor

Anil Kumar

Content

1. INTRODUCTION

Language is a method of communication with the help of which we can speak, read and write. Natural Language is that which normal Human beings use for communication in their day to day life. e.g. English ,Hindi ,French ,Russian etc. Computer Languages are C,C++,JAVA, Python etc. Natural language processing is a branch of data science that consists of systematic processes for analyzing, understanding, and deriving information from the text data in a smart and efficient manner. In this sense, we can say that Natural Language Processing (NLP) is the sub-field of Computer Science especially Artificial Intelligence (AI) that is concerned about enabling computers to understand and process human language.

There are two distinct focuses made in NLP, language processing and language generation. Language processing refers to analysis of language for the purpose of producing a meaningful interpretation, language generation refers to the task of generating natural language. Another distinction made in NLP is between analysis of spoken language and written language. Sentiment analysis is a subfield in NLP where the goal is to determine the attitude of a speaker or writer. Document-level sentiment classification is a functional task in sentiment analysis, and is crucial to understand user-generated content in social networks or product reviews. The task calls for identifying the overall sentiment polarity of a document.

As only less than 21 % data is structured and the remaining majority of this data exists in the textual form, which is highly unstructured in nature like tweets ,reviews messages etc., to get significant and actionable insights from text data, NLP is important. To solve a wide range of problems such as – automatic summarization, machine translation, named entity recognition, relationship extraction, sentiment analysis, speech recognition, and topic segmentation etc. natural language processing is done.

Models that are often used for sentiment analysis are Recurrent Neural networks and Long Short-Term networks. The objective of this study is to build a LSTM recurrent neural network model for classification of twitter sentiment.

2. SENTIMENT ANALYSIS

2.1. What is sentiment analysis ?

Sentiment Analysis is a field of Natural Language Processing responsible for systems that can extract opinions from natural language. NLP targets creating pipelines that can understand language like we humans do. Sentiment analysis or opinion mining is the computational study of people's opinions, sentiments, emotions, appraisals, and attitudes towards entities such as products, services, organizations, individuals, issues, events, topics, and their attributes.1 The inception and rapid growth of the field coincide with those of the social media on the Web, for example, reviews, forum discussions, blogs, micro-blogs, Twitter, and social networks, because for the first time in human history, we have a huge volume of opinionated data recorded in digital forms.

2.2. Types of sentiment analysis

2.2.1. Polarity Detection : Talking about the polarity of the sentence, that is, positive, negative or neutral. Sometimes the classification can be even more fine tuned, like very positive, positive, neutral, negative and very negative.

2.2.2. Emotion Detection : Detecting the emotion of the speaker from the sentence, for example, happy, sad, angry etc.

2.2.3. Intent Detection : Being able to detect not only what is present in the sentence but also its intent.

2.3. Phases of NLP

2.3.1. Morphological Processing

It is the first phase of NLP. The purpose of this phase is to break chunks of language input into sets of tokens corresponding to paragraphs, sentences and words. For example, a word like "uneasy" can be broken into two sub-word tokens as "un-easy".

2.3.2. Syntax Analysis

It is the second phase of NLP. The purpose of this phase is two folds: to check that a sentence is well formed or not and to break it up into a structure that shows the syntactic relationships between the different words. For example, sentences like “The school goes to the boy” would be rejected by syntax analyzer or parser.

2.3.3. Semantic Analysis

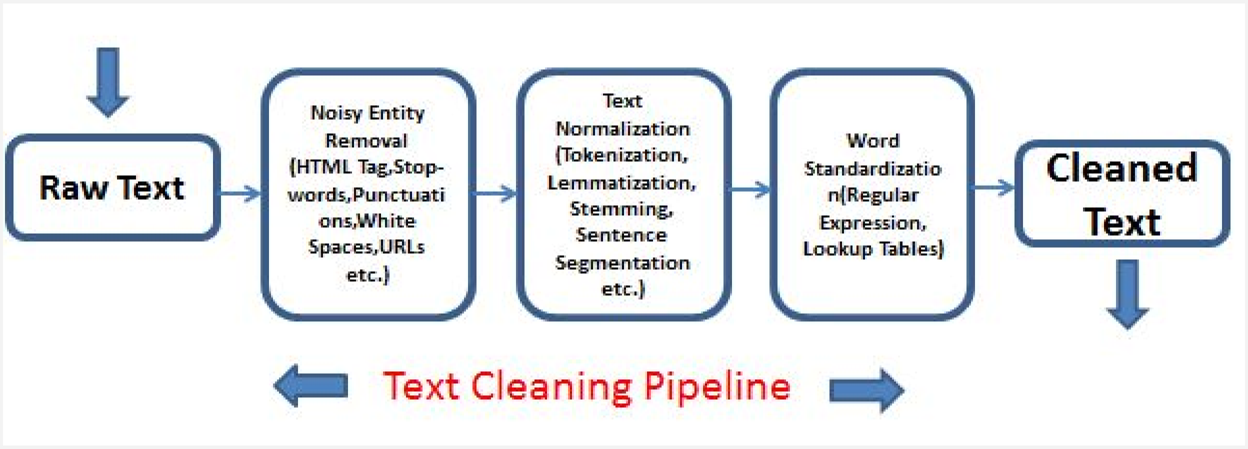
It is the third phase of NLP. The purpose of this phase is to draw exact meaning, or you can say dictionary meaning from the text. The text is checked for meaningfulness. For example, a semantic analyzer would reject a sentence like “Hot ice-cream”.

2.3.4. Pragmatic Analysis

It is the fourth phase of NLP. Pragmatic analysis simply fits the actual objects/events, which exist in a given context with object references obtained during the last phase (semantic analysis). For example, the sentence “Put the banana in the basket on the shelf” can have two semantic interpretations and pragmatic analyzer will choose between these two possibilities.

2.4. Basic Pipeline

The input provided to Sentiment analysis is not all useful. So cleaning is needed. After that feature extraction is done. After converting words to mathematical features, sentiment analysis is done by the Neural network using any suitable classification algorithm. .


Raw Text
Noisy Entity Removal (HTML Tag,Stop-words,Punctuations,White Spaces,URLs etc.)
Text Normalization (Tokenization, Lemmatization, Stemming, Sentence Segmentation etc.)
Word Standardization(Regular Expression, Lookup Tables)
Cleaned Text
Text Cleaning Pipeline

2.5. WORD EMBEDDING

Many deep learning models in NLP need word embedding results as input features.7 Word embedding is a technique for language modelling and feature learning, which transforms words in a vocabulary to vectors of continuous real numbers (e.g.,*word* "*ħat*" → (…, 0.15,…, 0.23,…, 0.41,…)). The technique normally involves a mathematical embedding from a high-dimensional sparse vector space (e.g., one-hot encoding vector space, in which each word takes a dimension) to a lower-dimensional dense vector space. Each dimension of the embedding vector represents a latent feature of a word. The vectors may encode linguistic regularities and patterns. One commonly used word embedding system is Word2Vec, which is essentially a computationally efficient neural network prediction model that learns word embeddings from text. It contains Continuous Bag-of-Words model (CBOW)13, and Skip-Gram model (SG). The CBOW model predicts the target word (e.g., “wearing”) from its context words (“the boy is _ a hat”, where “_” denotes the target word), while the SG model does the inverse, predicting the context words given the target word. Statistically, the CBOW model smoothens over a great deal of distributional information by treating the entire context as one observation. It is effective for smaller datasets. However, the SG model treats each context-target pair as a new observation and is better for larger datasets. Another frequently used learning approach is Global Vectorii (GloVe)17, which is trained on the nonzero entries of a global word-word co-occurrence matrix.

3. NEURAL NETWORK ARCHITECTURES FOR SENTIMENT ANALYSIS

There are mainly two kinds of neural network architectures, that can be mixed and matched feed-forward networks and Recurrent /Recursive networks. But it is known that Recurrent Neural Networks (RNN) are capable of dealing with dependencies in a sequence of data.So we will mainly go through RNN and Long Short Term memory(LSTM).

3.1. Recurrent Neural Network (RNN)

Recurrent Neural Network (RNN)22 is a class of neural networks whose connections between neurons form a directed cycle. Unlike feedforward neural networks, RNN can use its internal “memory” to process a sequence of inputs, which makes it popular for processing sequential information. The “memory” means that RNN performs the same task for every element of a sequence with each output being dependent on all previous computations, which is like “remembering” information about what has been processed so far.

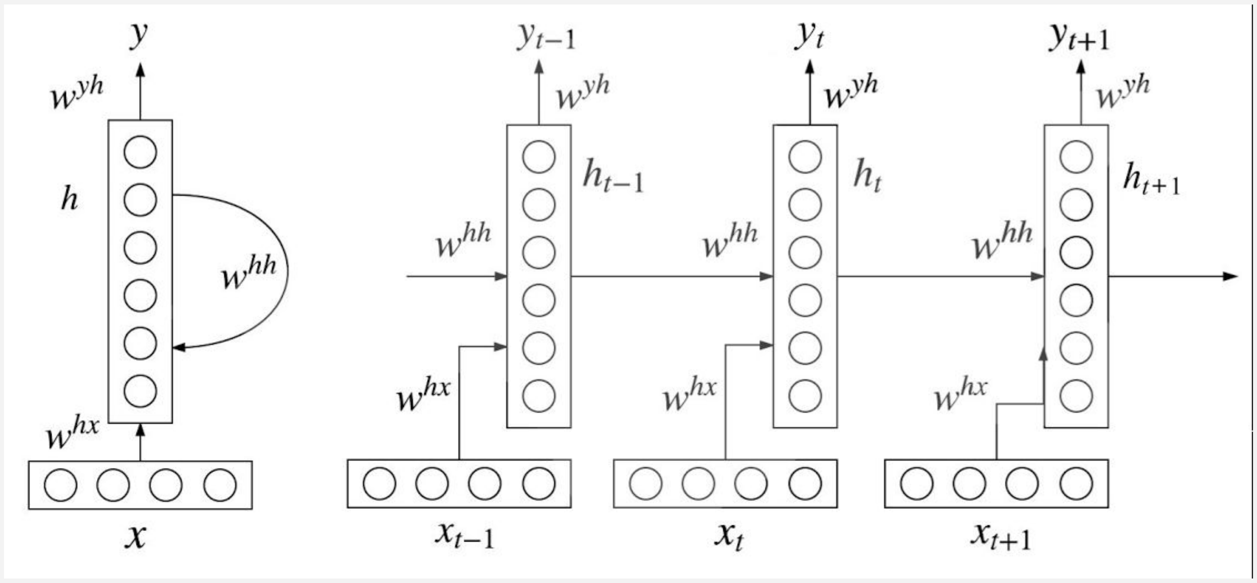


Fig. 2 Recurrent neural network

The figure above shows an example of a RNN. The left graph is an unfolded network with cycles, while the right graph is a folded sequence network with three time steps. The length of time steps is determined by the length of input. In Figure 2, x_t is the input vector at time step *t*. $\hbar_t$ is the hidden state at time step *t*, which is calculated based on the previous hidden state and the input at the current time step.

$$\hbar_t = f\,(w^{hh}\hbar_{t-1} + w^{hx}x_t)\text{...}(1)$$

In Equation (1), the activation function *f* is usually the tanh function or the ReLU function. w^{hx} is the weight matrix used to condition the input x_t . w^{hh} is the weight matrix used to condition the previous hidden state $\hbar_{t-1}$.

y_t is the output probability distribution over the vocabulary at step t. For example, if we want to predict the next word in a sentence, it would be a vector of probabilities across the word vocabulary.

$$y_t = softmax(\ w^{yh}h_t\ .)\text{..}\ (2)$$

The hidden state $\hbar_t$ is regarded as the memory of the network. It captures information about what happened in all previous time steps. y_t is calculated solely based on the memory $\hbar_t$ at time *t* and the corresponding weight matrix w^{yh}.

Unlike a feedforward neural network, which uses different parameters at each layer, RNN shares the same parameters (W^{hx},W^{hh},W^{yh}) across all steps. This means that it performs the same task at each step, just with different inputs. This greatly reduces the total number of parameters needed to learn. Theoretically, RNN can make use of the information in arbitrarily long sequences, but in practice, the standard RNN is limited to looking back only a few steps due to the vanishing gradient or exploding gradient problem. To deal with this problem LSTM models were developed and explained in the next section.

3.2. Long Short Term Memory network (LSTM)

Long Short Term Memory network (LSTM) is a special type of RNN, which is capable of learning long-term dependencies. All RNNs have the form of a chain of repeating modules. In standard RNNs, this repeating module normally has a simple structure. However, the repeating module for LSTM is more complicated. Instead of having a single neural network layer, there are four layers interacting in a special way. Besides, it has two states: hidden state and cell state.

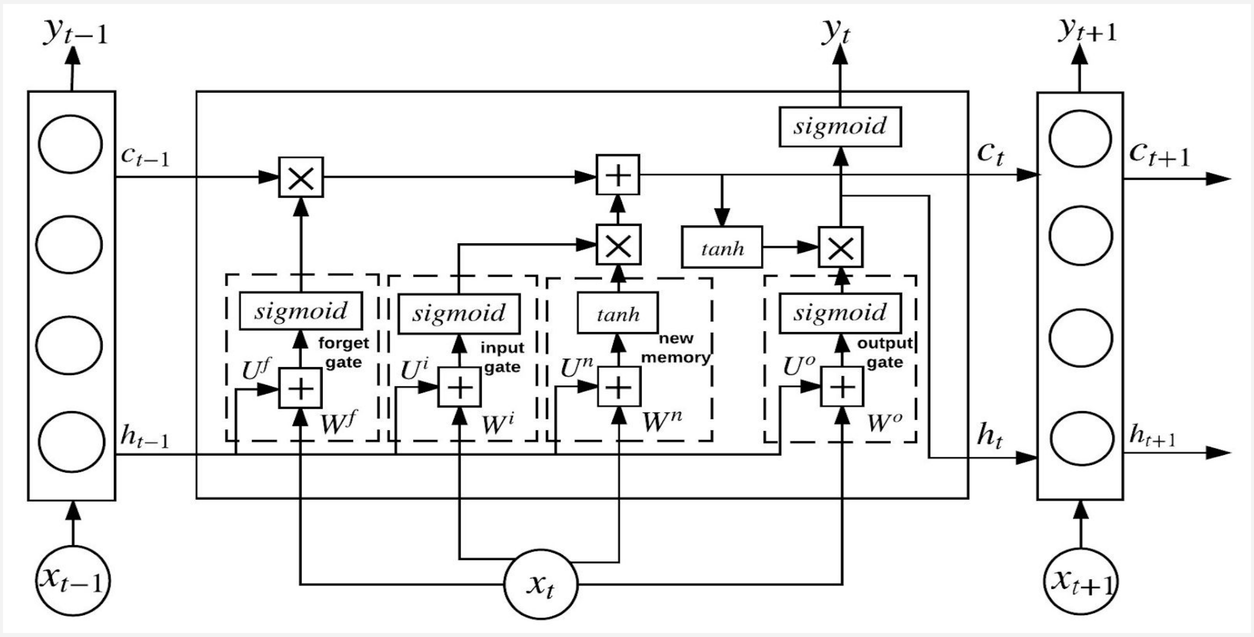


Fig 3 Long Short Term Memory network

Figure 3 shows an example of LSTM. At time step *t*, LSTM first decides what information to dump from the cell state. This decision is made by a sigmoid function/layer σ, called the "forget gate". The function takes $\hbar_{t-1}$ (output from the previous hidden layer) and x_t (current input), and outputs a number in [0, 1], where 1 means "completely keep" and 0 means "completely dump" in Equation (3).

$$f_t = \sigma\left(W^f x_t + U^f \hbar_{t-1}\right) \ldots\ldots (3)$$

Then LSTM decides what new information to store in the cell state. This has two steps. First, a sigmoid function/layer, called the "input gate" as Equation (4), decides which values LSTM will update. Next, a tanh function/layer creates a vector of new candidate values C_t , which will be added to the cell state. LSTM combines these two to create an update to the state.

$$i_t = \sigma\left(W^i x_t + U^i \hbar_{t-1}\right) \ldots\ldots (4)$$

$$\hat{C}_t = \tanh\left(W^n x_t + U^n \hbar_{t-1}\right) \ldots\ldots (5)$$

It is now time to update the old cell state C_{t-1} into new cell state C_t as Equation (6). Note that forget gate f_t can control the gradient passes through it and allow for explicit "memory" deletes and updates, which helps alleviate vanishing gradient or exploding gradient problem in standard RNN.

$$\hat{C}_t = f_t * C_{t-1} + i_t * \hat{C}_t \ldots\ldots (6)$$

Finally, LSTM decides the output, which is based on the cell state. LSTM first runs a sigmoid layer, which decides which parts of the cell state to output in Equation (7), called “output gate”. Then, LSTM puts the cell state through the tanh function and multiplies it by the output of the sigmoid gate, so that LSTM only outputs the parts it decides to as Equation (8).

$$o_t = \sigma\ (W^o x_t + U^o \hbar_{t-1}) \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (7)$$

$$\hbar_t = o_t * \tanh\ (C_t) \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (8)$$

LSTM is commonly applied to sequential data but can also be used for tree-structured data. Tai et introduced a generalization of the standard LSTM to Tree-structured LSTM (Tree-LSTM) and showed better performances for representing sentence meaning than a sequential LSTM. A slight variation of LSTM is the Gated Recurrent Unit (GRU). It combines the “forget” and “input” gates into a single update gate. It also merges the cell state and hidden state, and makes some other changes. The resulting model is simpler than the standard LSTM model, and has been growing in popularity.

4. OBJECTIVE AND METHODOLOGY

4.1. Objective

The objective of the study is to fit a LSTM Recurrent Neural Network model on Twitter dataset and determine the accuracy of the model and change in accuracy with change in hyperparameters and word Embedding in python. The other objective is to compare accuracy with the Multinomial Naive Bayes Classifier.

4.2. Methodology

First, we'll pass in words to an embedding layer. We need an embedding layer because we have tens of thousands of words, so we'll need a more efficient representation for our input data than one-hot encoded vectors. After input words are passed to an embedding layer, the new embeddings will be passed to LSTM cells. The LSTM cells will add recurrent connections to the network and give us the ability to include information about the sequence of words in the movie review data. Finally, the LSTM outputs will go to a sigmoid output layer. We're using a sigmoid function because we have two classes (0 is negative and 1 is non-negative) and a sigmoid will output predicted sentiment values between 0-1. Now we will see it stepwise.

Steps

1. Importing necessary libraries

   Numpy ,Pandas for data importing visualisation, matplotlib and seaborn libraries for plotting data, natural language toolkit library(nltk) for text processing, sklearn for data splitting data and accuracy assessment. Tensorflow, Keras for deep learning model building
2. Loading data

   Twitter data is loaded using pandas library.
3. Data preprocessing
   a. Defining sentiments based on score values
   b. Checking for duplicate records
   c. Checking Data consistency of Helpfulness Numerator and Helpfulness Denominator feature
   d. Checking for class imbalance

4. Text preprocessing
    a. Removing website links
    b. Removing html tags
    c. Decontracting(expanding from the original form)
    d. Removing the words with numeric digits
    e. Removing non-word characters
    f. Converting to lowercase
    g. Removing stop words
    h. Performing Lemmatization
5. Splitting into train and test set with 80:20 ratio
6. Model Building
    a. Fitting LSTM with Embedding layer

       Summary Model 1

```
 Model : "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 90, 128)           512000
_________________________________________________________________
spatial_dropout1d (SpatialDr (None, 90, 128)           0
_________________________________________________________________
lstm (LSTM)                  (None, 196)               254800
_________________________________________________________________
dropout (Dropout)            (None, 196)               0
_________________________________________________________________
dense (Dense)                (None, 100)               19700
_________________________________________________________________
dropout_1 (Dropout)          (None, 100)               0
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 202
=================================================================
Total params: 786,702 Trainable params: 786,702 Non-trainable params: 0
```

       The hyperparameters for this model are given as:

        1. Embedded dimension = 128

2. LSTM output shape = 196
3. Input length = length of training set
4. Spatial dropout = 0.5
5. activation function = ReLU
6. Final Activation function = softmax
7. epochs=10,
8. batch_size=50
9. Validation_split = 0.2

Summary: Model 2.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)               Output Shape            Param #
=================================================================
embedding_1 (Embedding)     (None, 100, 32)          192032
_________________________________________________________________
lstm_1 (LSTM)              (None, 100)             53200
_________________________________________________________________
dense_1 (Dense)            (None, 1)               101
=================================================================
Total params: 245,333
Trainable params: 245,333
Non-trainable params: 0
```

Hyperparameters used are

1. Embedding Vector length = 32
2. LSTM output shape = 100
3. Activation Function
4. No of epoch=8
5. batch_size=64
6. validation_split=0.2

b. Evaluating model performance on test data

The performance of the model is evaluated in terms of Training, validation and testing Accuracy.

Word Embedding Used = GloVe

Model

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 57, 25)            150000
_________________________________________________________________
spatial_dropout1d (SpatialDr (None, 57, 25)            0
_________________________________________________________________
bidirectional (Bidirectional (None, 57, 256)           157696
_________________________________________________________________
bidirectional_1 (Bidirection (None, 128)               164352
_________________________________________________________________
dropout (Dropout)            (None, 128)               0
_________________________________________________________________
dense (Dense)                (None, 1)                 129
=================================================================
Total params: 472,177
Trainable params: 472,177
Non-trainable params: 0
_________________________________________________________________
```

Multinomial Naive Bayes classifier is used for checking performance on using probabilistic model.

5. RESULTS AND DISCUSSION

The twitter dataset consists of a total of 99987 tweets. The dataset consists of a total of 56457 Positive sentiments and 43532 negative sentiments. The figure 1 shows the labeled sentiments of twitter dataset. The size of the dataset is sufficient for good classification. The split function separates data into 80:20 proportions. Training and test data consist of 79991 and 19998 tweets respectively..

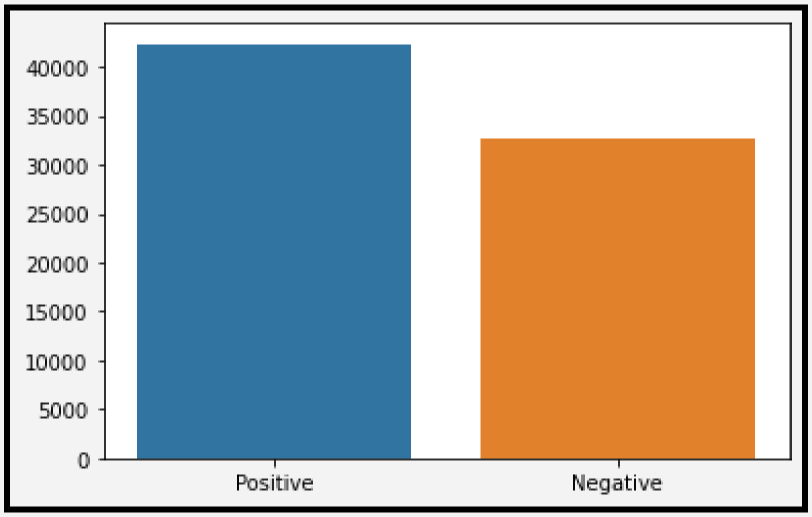
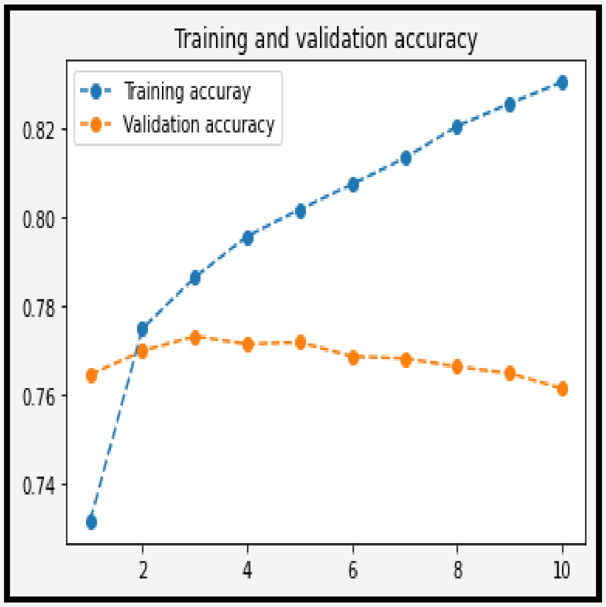


Figure.4 Barplot of twitter sentiment count

We build two LSTM models one with more number of layers and another with lesser number and for different values of hyper parameters and different activation functions. First we will analyse the Model 1. The figure 5 ang figure 6 shows the variation of training and validation accuracy with epochs and variation of training and validation loss with epochs. It is observed that as the number of epochs increases training accuracy increases and the validation accuracy reduces. And the opposite trends are observed in case of training and validation loss as it is obvious.

The overall training accuracy of the model is found to be **84.1157 %** with loss 0.35 and the testing accuracy is found to be **76.19** % with loss of 0.51. And the confusion matrix is shown in figure 7.



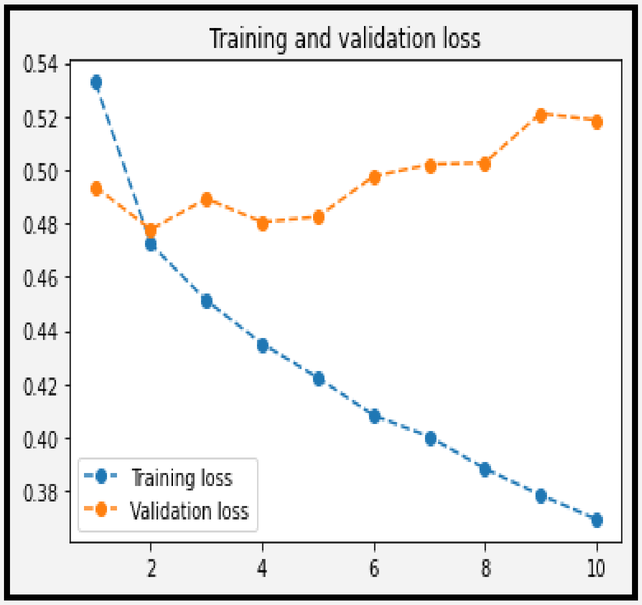


Figure 5 Training and Validation accuracy

Figure 6 Training and validation loss

The confusion matrix from prediction over test data shows the classification and misclassification done by the datta. Figure 7 shows that 5928 negative tweets and 9309 positive are correctly classified by the LSTM RNN model. 1983 positive sentiments are wrongly classified as negative and 2778 negative tweets are wrongly classified as positive.

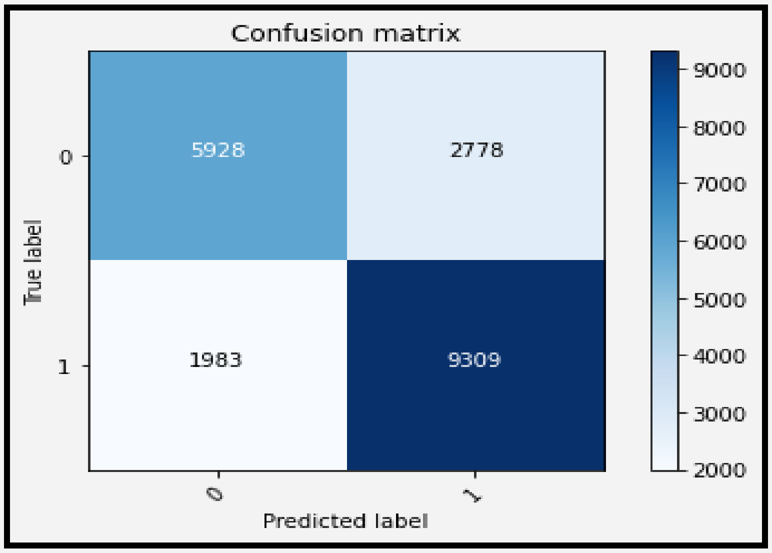


Figure. 7 Confusion Matrix

Model 2 Output

In the similar way LSTM with different hyperparameter performance is determined. The figure 8 and 9 shows variation of training accuracy, loss and validation accuracy, loss with epochs. The model achieves higher training accuracy upto 93 % . but validation accuracy decreases down upto 70%. In the opposite manner losses vary.

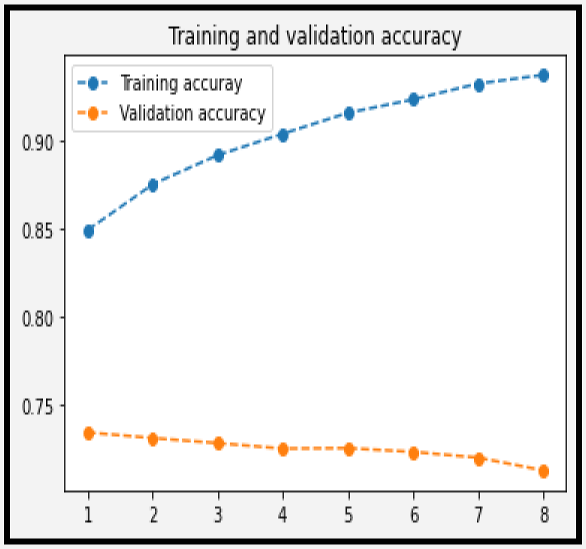


Figure 8 Training and Validation accuracy

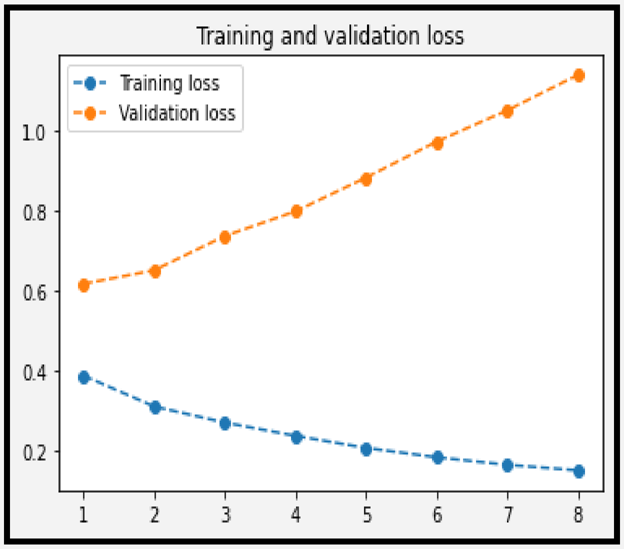


Figure 9 Training and validation loss

The model achieves overall training accuracy of **90.26 %** test accuracy of **70.77 %.** And the confusion matrix on the test dataset shows that the model predicts 5649 negative and 8504 positive sentiments correctly. And misclassified 2744 positive sentiments as negative and 3101 negative as positive sentiment.

On using **GloVe** word embedding in **LSTM** achieved accuracy of **74.90%** on test data

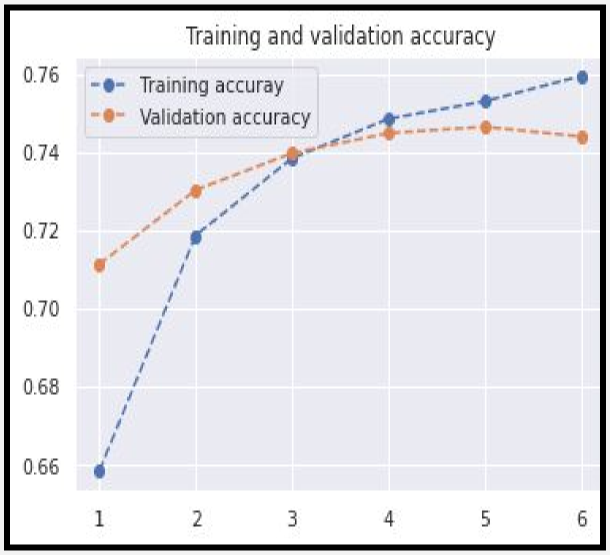


Figure 10 Training and Validation accuracy

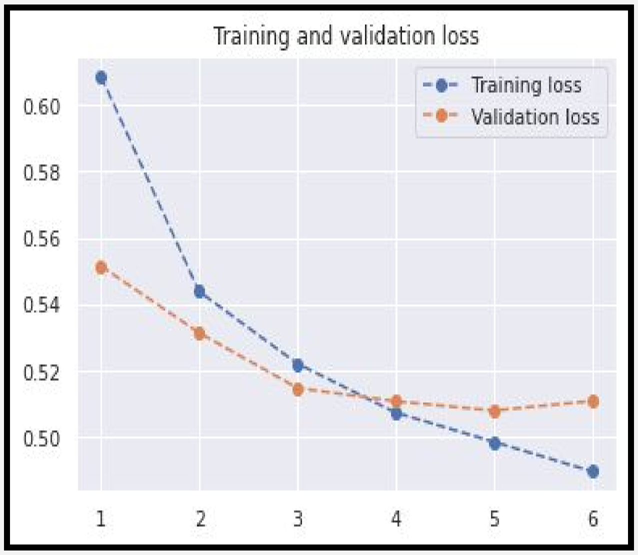
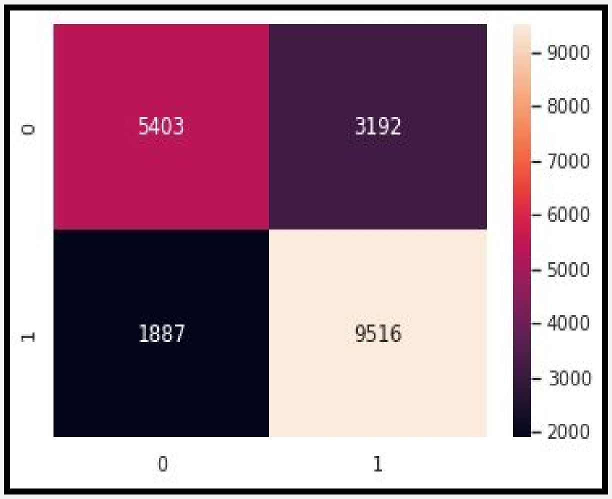


Figure 11 Training and validation loss



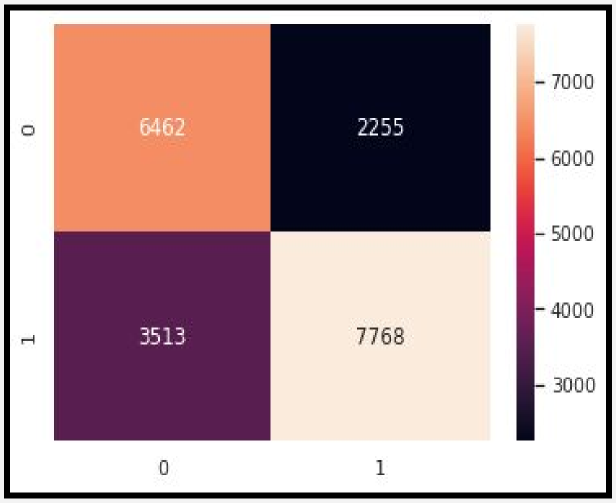


Figure 12, Confusion Matrix for LSTM

Figure 13, Confusion Matrix for Naive Bayes

With Multinomial Naive Bayes classifier achieved **Accuracy of 71.16% and F1 Score: 72.93**.

6. CONCLUSION

Sentiment classification is a task in NLP where documents are classified according to their sentiment. Models that are often used for sentiment analysis are RNN and LSTM. RNNs are neural networks with loops in them, allowing the network to have a memory. Having a memory in a network is useful because when dealing with sequenced data such as text, the meaning of a word depends on the context of the previous text. One shortcoming of the RNN is that it is only capable of dealing with short-term dependencies. LSTM is a network which addresses this issue by introducing a long-term memory in the network. In an LSTM network sequenced data is processed by using gate vectors at each position to control the passing of information along the sequence.In Compare to Multinomial Naive Bayes classifier Long Short Term memory RNN provides good accuracy.

The accuracy of classification largely depends on the choice of hyperparameters used for building the model. The cleaning of data and text processing are the one of the most important operations. This also contributes to the better accuracy of the model and also reduces the feature..

7. REFERENCES

1. A Primer on Neural Network Model For Natural Language Processing by Yoav Goldberg.
2. https://www.analyticsvidhya.com/blog/2017/01/ultimate-guide-to-understand-implement-natural-language-processing-codes-in-python/
3. https://towardsdatascience.com/recurrent-neural-networks-by-example-in-python-ffd204f99470
4. https://towardsdatascience.com/sentiment-analysis-concept-analysis-and-applications-6c94d6f58c17
5. https://towardsdatascience.com/sentiment-analysis-using-lstm-step-by-step-50d074f09948
6. Deep Learning for Sentiment Analysis: A Survey by Zhang lie
7. Understanding LSTM Networks, http://colah.github.io/posts/2015-08-Understanding-LSTMs/ .