# Ahsanullah University of Science and Technology

## Department of Computer Science and Engineering

| | | |
|---|---|---|
| **Course No.** | **:** | CSE4238 |
| **Course Name** | **:** | Soft Computing Lab |

**Assignment No. :** 03

**Submitted By:**

| | | |
|---|---|---|
| Name | : | Shahriar Hasan Chowdhury |
| ID No. | : | 17 01 04 030 |
| Session | : | Fall - 2020 |
| Section | : | A (A2) |

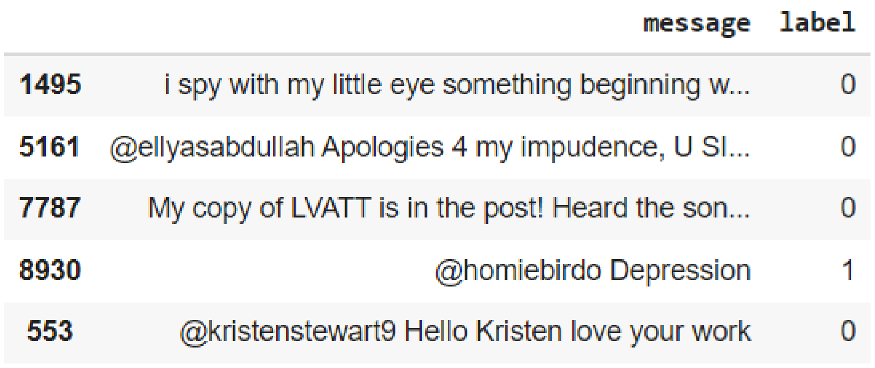
## Table of Contents

## Introduction

The task of this experiment is to build a sequential model that can classify sentiments form given dataset. As the last three digits of my id is 030, so as per the condition (030 % 3 == 0) I have to use dataset 1, and for the model ((030 + 1) % 5 == 1) i have to use LSTM stacked with two recurrent layers. The purpose of this assignment is to learn how to deal with sequential data like text, also how to preprocess text data and apply to different model.

## Dataset

As mentioned earlier, I have to use dataset 1 to apply LSTM model. The following dataset contains two column (message, label). The message column is tweets from twitter and label is the sentiment. There are two unique label in the label column. 0 means negative and 1 means positive sentiments.



| | message | label |
|---|---|---|
| 1495 | i spy with my little eye something beginning w... | 0 |
| 5161 | @ellyasabdullah Apologies 4 my impudence, U Sl... | 0 |
| 7787 | My copy of LVATT is in the post! Heard the son... | 0 |
| 8930 | @homiebirdo Depression | 1 |
| 553 | @kristenstewart9 Hello Kristen love your work | 0 |

As we can see the datasets contains lots of unnecessary information like mentions stop words, links, punctuations, special characters etc. we have to remove them before applying it to the model in other words we have to do some preprocessing on the message column.

## Preprocessing

### Removing mentions, links, tags

First step for preprocessing is to remove mentions, links and tags from the message column. For this we have used python regular expression to clean the message column.

```
def removing_mentions(text):
    text = re.sub(r'@[A-Za-z0-9]+', '', text)      # removing @mentions
    text = re.sub(r'@[A-Za-zA-Z0-9]+', '', text)   # removing @mentions
    text = re.sub(r'@[A-Za-z]+', '', text)         # removing @mentions
    text = re.sub(r'@[-)]+', '', text)             # removing @mentions

    return text

def removeing_links(text):
    text = re.sub(r'https?\/\/\S+', '', text)      # removing the hyper link
    text = re.sub(r'http?\/\/\S+', '', text)      # removing the hyper link

    return text

def removing_tags(text):
    text = re.sub(r'#', '', text )               # removing '#' sign
    text = re.sub(r'RT[\s]+', '', text)          # removing RT
    text = re.sub(r'&[a-z;]+', '', text)         # removing '&gt;'

    return text
```

## Removing Stop words

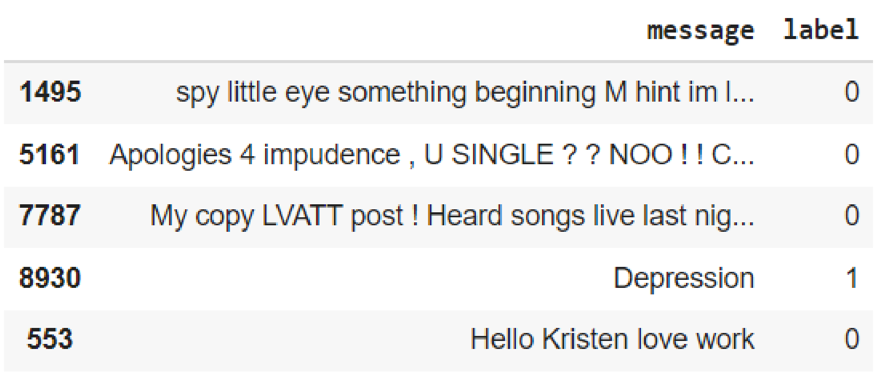
One of the most important step of preprocessing text is to remove stop words. Stop words are those words that do not necessarily carry any meaningful information to the sentence. By removing stop words the length of the sentence can be reduced which will reduce the number of calculation for predicting sentiments. I have used nltk library to import English stop words.

```
nltk.download('stopwords')
stopwords = nltk.corpus.stopwords.words('english')

def remove_stopwords(text):
    tokenizer = BasicTokenizer()
    words = tokenizer.tokenize(text)
    filtered_words = [w for w in words if not w in stopwords]

    str = " "
    return str.join(filtered_words)
```

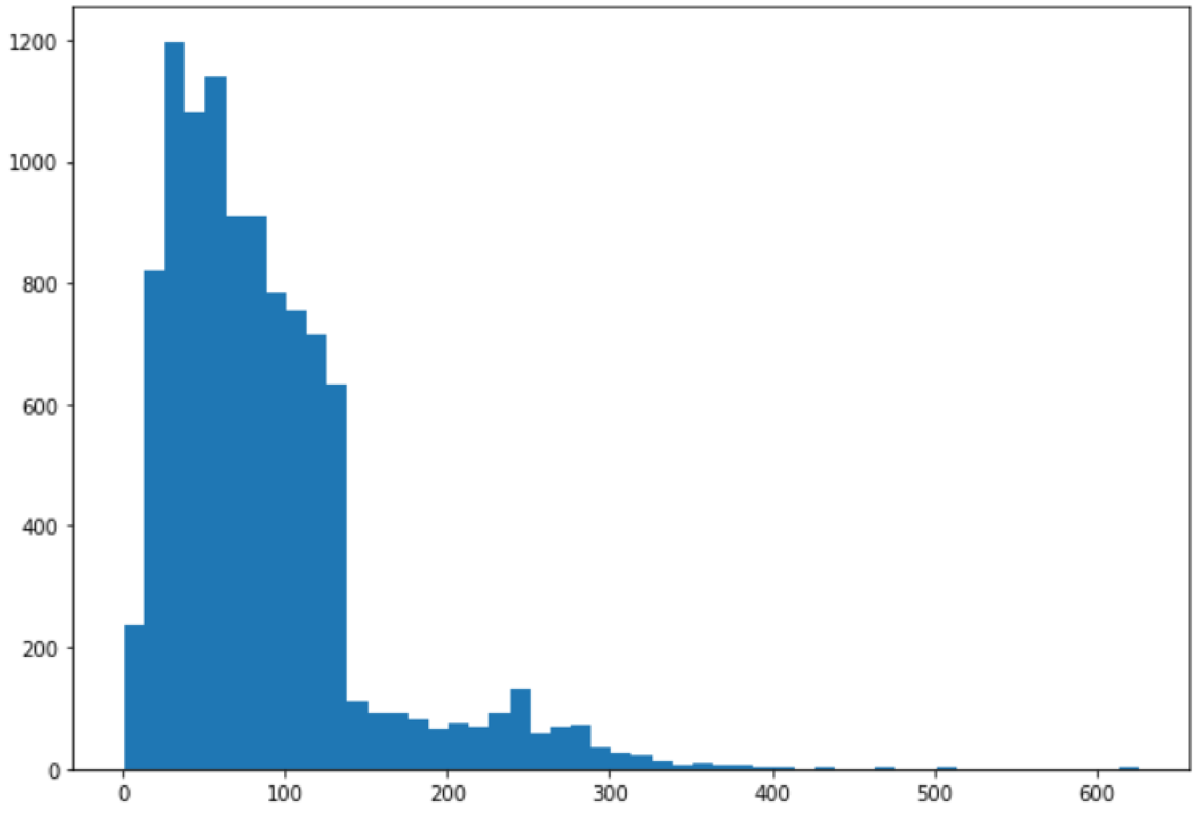
After preprocessing the datasets looks like this,



| | message | label |
|---|---|---|
| 1495 | spy little eye something beginning M hint im l... | 0 |
| 5161 | Apologies 4 impudence , U SINGLE ? ? NOO ! ! C... | 0 |
| 7787 | My copy LVATT post ! Heard songs live last nig... | 0 |
| 8930 | Depression | 1 |
| 553 | Hello Kristen love work | 0 |

## Vectorization

After preprocessing step, we need to convert the textual data into some form of number. This step is called text to vectorization. Is the message column each message is of different lengths. Deep learning model can not works with different length of data. So I made another column "length" to see the length of each message and plot a histogram of it.

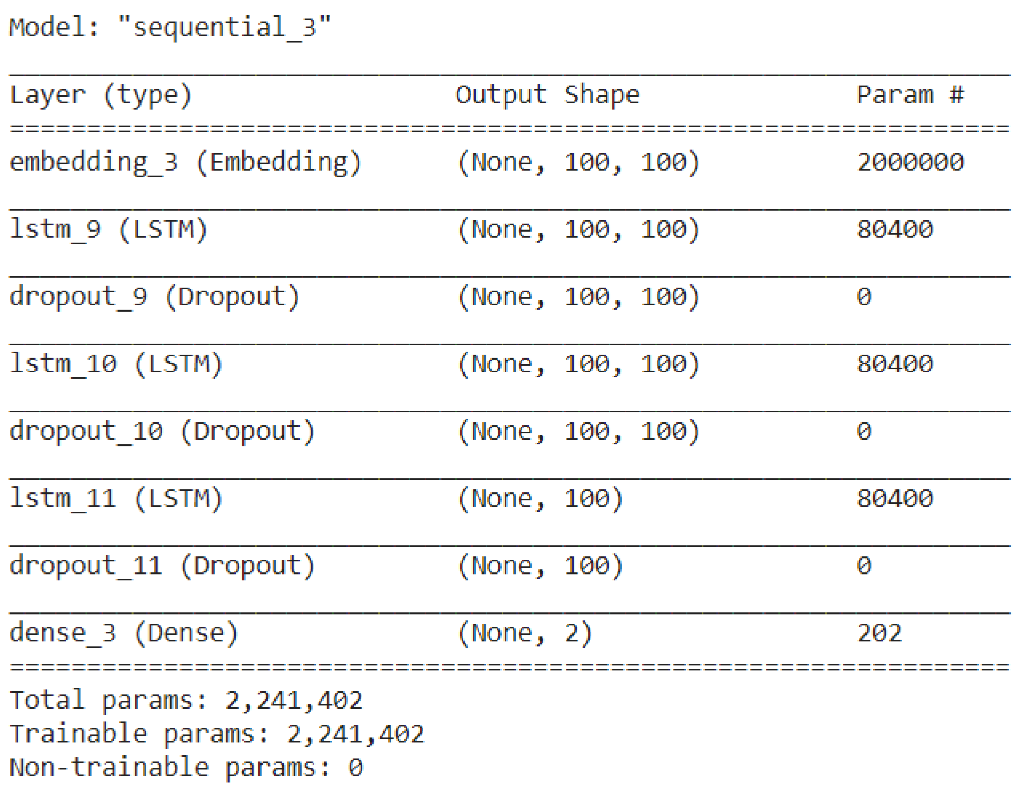


From the figure we can see there are different lengths of message. And the maximum length message is little more than 600. But the amount of that is very small. If we take a length of 600 that will increase unnecessary calculation, as most of the message will have padding of 0. So I have taken a fixed length of 100 per each message.

Also there are a total of 18,175 unique tokens in the dataset. So I have taken the maximum word length of 20,000. After that I tokenize each message and gave them a unique number. After that I have applied padding in the begging of the message if needed, and made train test data. The shape of the training data is (8251, 100). And the shape of the testing data is (2063, 100).

## Model Building

As mentioned earlier I have to use LSTM model. The first layer is an embedding layer where the embedding dimension is 100. Then I have used three LSTM layers. After each LSTM layers I have added dropout layer to prevent overfitting the model. The summary of the model is given below.



```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_3 (Embedding)      (None, 100, 100)          2000000
_________________________________________________________________
lstm_9 (LSTM)                (None, 100, 100)          80400
_________________________________________________________________
dropout_9 (Dropout)          (None, 100, 100)          0
_________________________________________________________________
lstm_10 (LSTM)               (None, 100, 100)          80400
_________________________________________________________________
dropout_10 (Dropout)         (None, 100, 100)          0
_________________________________________________________________
lstm_11 (LSTM)               (None, 100)               80400
_________________________________________________________________
dropout_11 (Dropout)         (None, 100)               0
_________________________________________________________________
dense_3 (Dense)              (None, 2)                 202
=================================================================
Total params: 2,241,402
Trainable params: 2,241,402
Non-trainable params: 0
```

I have used tanh activation function for the LSTM layers and as it is a binary classification problem I have used sigmoid activation function in the dense layer which is the output layer that has two output.

As this is a small dataset I have used 10 epoch and batch size 128. The summary of each epoch is given below.

```
Epoch 1/15
65/65 [==============================] - 9s 75ms/step - loss: 0.4099 - accuracy: 0.8387
Epoch 2/15
65/65 [==============================] - 5s 73ms/step - loss: 0.0763 - accuracy: 0.9771
Epoch 3/15
65/65 [==============================] - 5s 73ms/step - loss: 0.0115 - accuracy: 0.9977
Epoch 4/15
65/65 [==============================] - 5s 73ms/step - loss: 0.0049 - accuracy: 0.9993
Epoch 5/15
65/65 [==============================] - 5s 73ms/step - loss: 0.0044 - accuracy: 0.9995
Epoch 6/15
65/65 [==============================] - 5s 73ms/step - loss: 0.0028 - accuracy: 0.9995
Epoch 7/15
65/65 [==============================] - 5s 73ms/step - loss: 0.0025 - accuracy: 0.9996
Epoch 8/15
65/65 [==============================] - 5s 73ms/step - loss: 0.0021 - accuracy: 0.9996
Epoch 9/15
65/65 [==============================] - 5s 73ms/step - loss: 0.0065 - accuracy: 0.9992
Epoch 10/15
65/65 [==============================] - 5s 73ms/step - loss: 0.0035 - accuracy: 0.9995
Epoch 11/15
65/65 [==============================] - 5s 73ms/step - loss: 0.0030 - accuracy: 0.9994
Epoch 12/15
65/65 [==============================] - 5s 73ms/step - loss: 0.0014 - accuracy: 0.9998
Epoch 13/15
65/65 [==============================] - 5s 73ms/step - loss: 0.0027 - accuracy: 0.9994
Epoch 14/15
65/65 [==============================] - 5s 73ms/step - loss: 0.0014 - accuracy: 0.9998
Epoch 15/15
65/65 [==============================] - 5s 72ms/step - loss: 0.0013 - accuracy: 0.9998
```

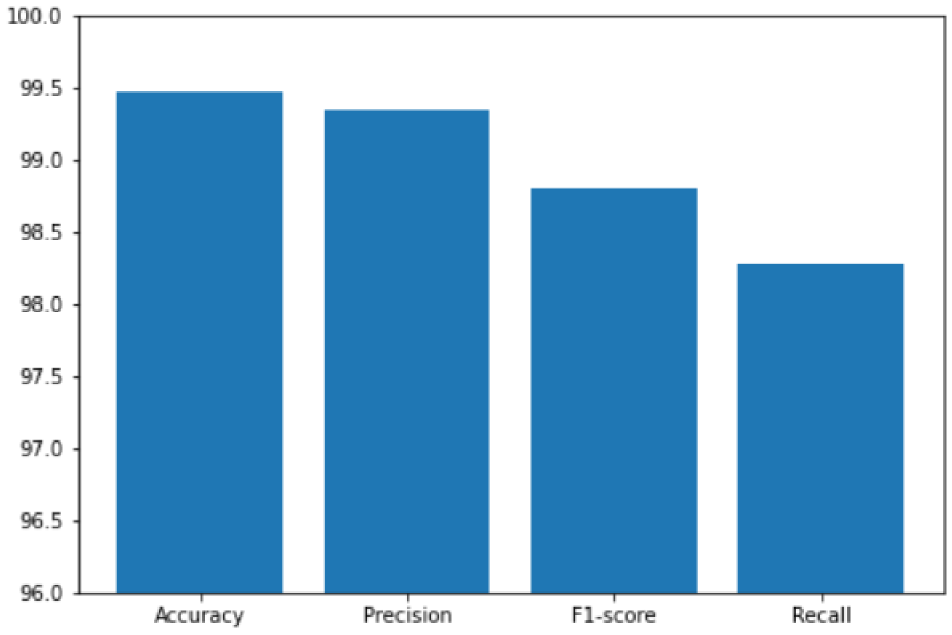
## Performance Evaluation

For evaluating the model result I have showed accuracy, precision, f1-score and recall. The performance of the model is given bellow.

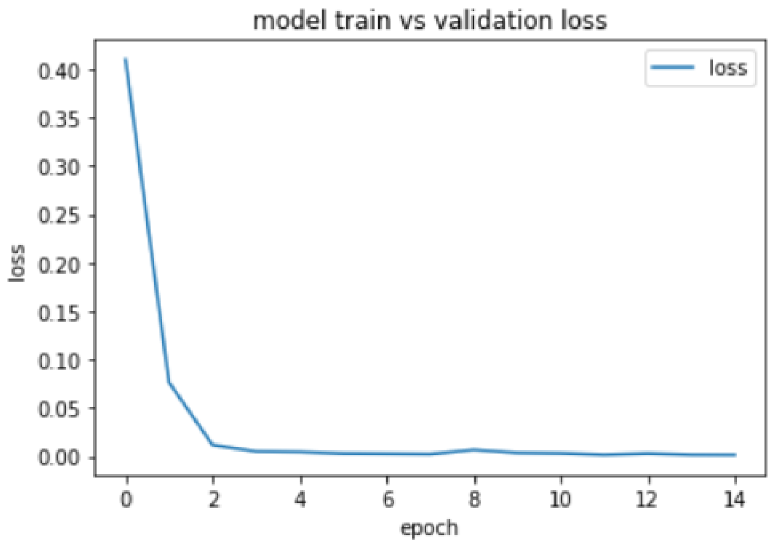
### Performance Table

| | |
|---|---|
| **Accuracy** | **99.47** |
| **Precision** | **99.34** |
| **F1-score** | **98.80** |
| **Recall** | **98.27** |

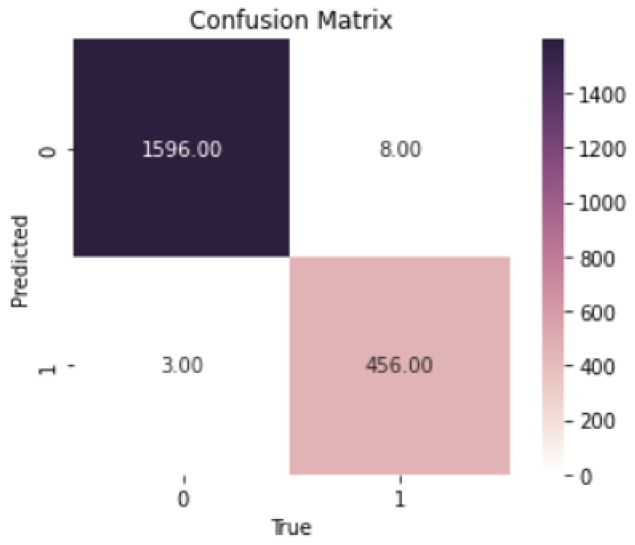
## Comparison chart



## Loss Curve



## Confusion Matrix



## Conclusion

From the above section we can see that the performance of the model is excellent. Also we have learnt to work with textual dataset and Long Short Term Memory model to do binary classification. Also we have learnt to preprocess text data and convert it to a vector so that we can use it to predict sentiments.

## Code File

All the codes are uploaded to GitHub. Chick here to visit GitHub repository.