DEEP LEARNING CS6005

Mini Project on Natural Language Processing

Sentiment Analysis on Tweets using LSTM

Date : 20-05-2021

Praveen RS

2018103577

P Batch

Problem Statement

This project covers the sentiment analysis of any topic by parsing the tweets fetched from Twitter using Python. **Sentiment Analysis** is the process of computationally identifying and categorizing opinions expressed in a piece of text, especially in order to determine whether the writer's attitude towards a particular topic, product, etc. is positive, negative, or neutral. We perform sentiment analysis for the following purposes:

Business: In marketing field companies use it to develop their strategies, to understand customers' feelings towards products or brand, how people respond to their campaigns or product launches and why consumers don't buy some products.

Politics: In political field, it is used to keep track of political view, to detect consistency and inconsistency between statements and actions at the government level. It can be used to predict election results as well!

Public Actions: Sentiment analysis also is used to monitor and analyse social phenomena, for the spotting of potentially dangerous situations and determining the general mood of the blogosphere

This Sentiment Analysis comes under the domain of Natural Language Processing.

Natural Language Processing

Natural language processing strives to build machines that understand and respond to text or voice data—and respond with text or speech of their own—in much the same way humans do.

Natural Language Processing (NLP) refers to AI method of communicating with an intelligent system using a natural language such as English.

Processing of Natural Language is required when you want an intelligent system like robot to perform as per your instructions, when you want to hear decision from a dialogue based clinical expert system, etc.

Natural language processing includes many different techniques for interpreting human language, ranging from statistical and machine learning methods to rules-based and algorithmic approaches. We need a broad array of approaches because the text- and voice-based data varies widely, as do the practical applications.

Basic NLP tasks include **tokenization** and parsing, **lemmatization/stemming**, part-of-speech tagging, language detection and identification of **semantic relationships**.

In this project, we use an LSTM Network to perform sentiment analysis on the twitter dataset.

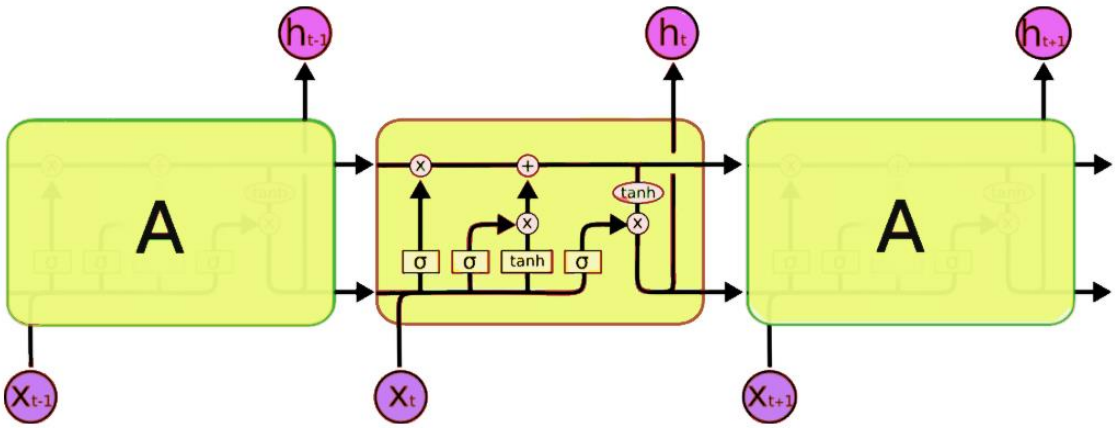
LSTM

With the recent breakthroughs that have been happening in data science, it is found that for almost all of these sequence prediction problems, Long Short Term Memory networks, a.k.a. LSTMs have been observed as the most effective solution. LSTMs have an edge over conventional feed-forward neural networks and RNN in many ways. This is because of their property of selectively remembering patterns for long durations of time.

In order to add a new information, RNN (Recurrent Neural Networks) transforms the existing information completely by applying a function. Because of this, the entire information is modified, on the whole, i. e. there is no consideration for *'important'* information and *'not so important'* information.

LSTMs on the other hand, make small modifications to the information by multiplications and additions. With LSTMs, the information flows through a mechanism known as cell states. This way, LSTMs can selectively remember or forget things. The information at a particular cell state has three different dependencies.

Architecture of LSTMs



A typical LSTM network is comprised of different memory blocks called **cells** (the rectangles that we see in the image above)**.** There are two states that are being transferred to the next cell; the **cell state** and the **hidden state**. The memory blocks are responsible for remembering things and manipulations to this memory is done through three major mechanisms, called **gates.**

We will use the python library **Keras** for implementing LSTM, which is a high-level API for neural networks and works on top of TensorFlow or Theano.

Dataset Details

Our dataset consists of a large number of tweets and their corresponding user info. The target attribute of our dataset is the Sentiment which says the semantic meaning involved in the tweet. Here, we use over 50,000 tweets to train our model.

	A	B	C	D	E	F	G	H	I	J
1	Fri Mar 23 00:40:32 +0000 2018	RT @ALXTOKEN: Paul Krugman,	myresumerocket	16522	0	[]	<a href="http://twitter.com" rel="nof	['neutral']		
2	Fri Mar 23 00:40:34 +0000 2018	@lopp @_Kevin_Pham @psych	BitMocro	1295	0	[u'Bitcoin']	<a href="http://twitter.com/downloa	['neutral']		
3	Fri Mar 23 00:40:35 +0000 2018	RT @tippereconomy: Another u	hojachotopur	6090	0	[u'blockchain', u'Tipper', u'Tipp	<a href="http://twitter.com" rel="nof	['positive']		
4	Fri Mar 23 00:40:36 +0000 2018	free coins https://t.co/DiuoePJ	denies_distro	2626	0	[]	<a href="http://twitter.com" rel="nof	['positive']		
5	Fri Mar 23 00:40:36 +0000 2018	RT @payvxofficial: WE are	aditzgraha	184	0	[]	<a href="http://twitter.com/downloa	['positive']		
6	Fri Mar 23 00:40:36 +0000 2018	Copy successful traders	VictorS61164810	14	0	[]	<a href="http://twitter.com" rel="nof	['positive']		
7	Fri Mar 23 00:40:37 +0000 2018	RT @bethereumteam: We're	ClarkKalel4	44	0	[u'surprise', u'presents', u'crypt	<a href="http://twitter.com/downloa	['positive']		
8	Fri Mar 23 00:40:37 +0000 2018	One click to start mining cryptoc	cloud_speaker	6560	0	[u'bitcoin', u'PaaS', u'cloudnetw	<a href="http://itunes.apple.com/us/	['neutral']		
9	Fri Mar 23 00:40:38 +0000 2018	first speaker @digitsu	MADinMelbourne	2991	0	[]	<a href="http://twitter.com" rel="nof	['positive']		
10	Fri Mar 23 00:40:40 +0000 2018	@p0nd3ea Bitcoin wasn't built t	APompliano	96897	0	[]	<a href="http://twitter.com" rel="nof	['positive']		
11	Fri Mar 23 00:40:40 +0000 2018	@historyinflicks Buddy if I had v	martsendo	654	0	[]	<a href="http://twitter.com" rel="nof	['positive']		
12	Fri Mar 23 00:40:40 +0000 2018	RT @livemarketcap: ICE Agency	NewsFeedDE	472	0	[u'Regulation', u'AML', u'arrests	<a href="https://twitter.com/NewsFe	['neutral']		
13	Fri Mar 23 00:40:41 +0000 2018	RT @Vaultoro: We have all seer	FinancialMinds_	11	0	[]	<a href="http://twitter.com/downloa	['neutral']		
14	Fri Mar 23 00:40:42 +0000 2018	@eatBCH @Bitcoin @signalapp	silveragorism	560	0	[]	<a href="http://twitter.com/downloa	['neutral']		
15	Fri Mar 23 00:40:42 +0000 2018	RT @_bitcoinmonster: Bitcoin	denies_distro	2626	0	[]	<a href="http://twitter.com" rel="nof	['positive']		
16	Fri Mar 23 00:40:43 +0000 2018	RT @SteveRichFXCorp: #Breakir	iminifintech	259	0	[u'Breaking', u'SteveRichFXCorp	<a href="https://www.oxoscorp.com'	['positive']		
17	Fri Mar 23 00:40:44 +0000 2018	RT @bethereumteam: This	calogerojonie51	572	0	[u'milestone']	<a href="http://twitter.com" rel="nof	['positive']		
18	Fri Mar 23 00:40:43 +0000 2018	Copy successful traders	amymiller916ud	18	0	[]	<a href="http://twitter.com" rel="nof	['positive']		
19	Fri Mar 23 00:40:44 +0000 2018	Applicature Joins the Ethereum	PureCryptoNews	55	0	[u'Cryptocurrency', u'Crypto', u'	<a href="https://ifttt.com" rel="nofol	['neutral']		
20	Fri Mar 23 00:40:45 +0000 2018	RT @neolite_token: $NEOLite	XPrience1	110	0	[u'AIRDROP']	<a href="http://twitter.com/downloa	['neutral']		
21	Fri Mar 23 00:40:45 +0000 2018	First to USD $0.10	ballstreet_	1403	0	[u'tron', u'bitcoin', u'trader', u'tr	<a href="http://twitter.com/downloa	['positive']		
22	Fri Mar 23 00:40:47 +0000 2018	RT @payvxofficial: Support Pay	crassquit	730	0	[]	<a href="http://twitter.com" rel="nof	['neutral']		
23	Fri Mar 23 00:40:47 +0000 2018	Purchase RSA 2048 bit cracking s	theautomatski	70779	0	[]	<a href="http://automatski.com" rel=	['neutral']		

Our dataset of tweets focuses only on **Bitcoin**. Bitcoin is a decentralized digital currency, without a central bank or single administrator, that can be sent from user to user on the peer-to-peer bitcoin network without the need for intermediaries. Bitcoins are created as a reward for a process known as mining. They can be exchanged for other currencies, products, and services.

The tweets have #Bitcoin and #btc hashtag. The tweets were extracted from Open-source APIs available from the Internet. These tweets can also be extracted from ‘tweepy’, a Python library.

Modules Used

1.Loading the dataset and necessary libraries

Firstly, the corresponding '.csv' file is loaded into the program. Then, all the necessary libraries are imported for the execution of the program and visualization of the results.

2. Dataset Pre-processing

We can see the columns are without any proper names. Let's rename them for our references. We are going to train only on text to classify its sentiment. So, we can ditch the rest of the useless columns. We will find the length of the tweets and add them as a separate column. Tweet texts often consists of other user mentions, hyperlink texts, emoticons and punctuations. In order to use them for learning using a Language Model. We cannot permit those texts for training a model. So, we have to clean the text data using various pre-processing and cleansing methods. Hence, we are going to clean up the tweets, remove special chars, stop words, URL links, etc. This cleansed dataset is split into training and testing datasets, with test data being 20% of the whole dataset.

The tweets can be in the form of really long sentences and paragraphs, hence we tokenize them into lists of words.

3. Word Embedding

Word Embedding is one of the popular representations of document vocabulary. It is capable of capturing context of a word in a document, semantic and syntactic similarity, relation with other words, etc. Basically, it's a feature vector representation of words which are used for other natural language processing applications.

We should feed in a sequence of numbers to it and also we should ensure there is no variance in input shapes of sequences. It all should be of same length. But texts in tweets have different count of words in it. To avoid this, we seek a little help from pad_sequence to do our job before word embedding. It will make all the sequence in one constant length.

4. Model Creation and Training

We are clear to build our Deep Learning model. While developing a DL model, we should keep in mind of key things like Model Architecture, Hyperparmeter Tuning and Performance of the model.

For model architecture, we use

1) Embedding Layer - Generates Embedding Vector for each input sequence.

2) Conv1D Layer - Its using to convolve data into smaller feature vectors.

3) Maxpooling Layer – Size (2,2) and stride of (2,2) is used. Pool_size of 2 is used.

3) LSTM - Long Short Term Memory, its a variant of RNN which has memory state cell to learn the context of words which are at further along the text to carry contextual meaning rather than just neighbouring words as in case of RNN. This layer has 50 units and a Dropout of 0.2.

4) Dense - Fully Connected Layers for final classification of the tweet.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 30, 50)            1000000
_________________________________________________________________
conv1d (Conv1D)              (None, 30, 32)            4832
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 15, 32)            0
_________________________________________________________________
lstm (LSTM)                  (None, 50)                16600
_________________________________________________________________
dense (Dense)                (None, 3)                 153
=================================================================
Total params: 1,021,585
Trainable params: 1,021,585
Non-trainable params: 0
_________________________________________________________________
None
```

For training the model, we have to set appropriate values to a few parameters:

Optimizer	Adam
Loss	Categorical Cross-entropy
Epochs	10
Batch size	128
Learning rate	0.01
Monitor	Validation loss

We will also be using early-stopping with a patience of 2 to avoid overfitting.

5.Result Analysis

The model's accuracy vs epochs graph and loss vs epochs graph are plotted for both training data and validation data. Since we don't get a categorical value from the result, we index the three classes as numerical labels from 0 to 2. Classification report is generated for the test data which contains information about precision, recall and f1-score, and the confusion matrix is plotted with the heatmap for visualization.

Source Code

Importing the necessary libraries and datasets

[1]

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

from nltk.corpus import stopwords
import re

from sklearn.model_selection import train_test_split
from keras.preprocessing.text import Tokenizer
from keras.preprocessing import sequence

from keras.models import Sequential
from keras.layers import Dense,Embedding,Conv1D,MaxPooling1D,LSTM
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
from keras.callbacks import EarlyStopping
from keras.optimizers import Adam
```

Dropping the unnecessary attributes

[2]

```
# Load Data
df = pd.read_csv('bitcointweets.csv', header=None)
df = df[[1,7]]
df.columns = ['tweet','label']
df.head()
```

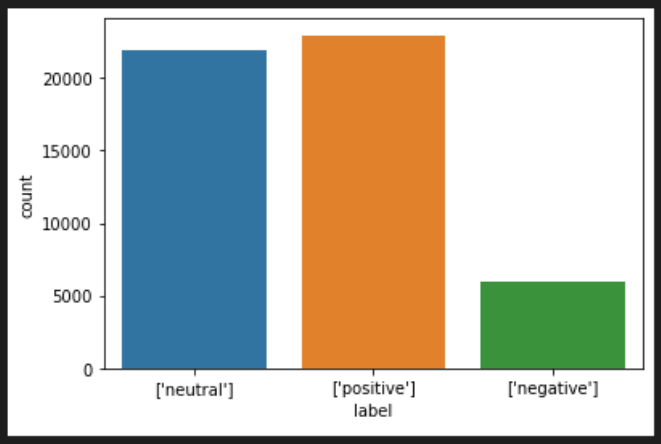
	tweet	label
0	RT @ALXTOKEN: Paul Krugman, Nobel Luddite. I h...	['neutral']
1	@lopp @_Kevin_Pham @psycho_sage @naval But @Pr...	['neutral']
2	RT @tippereconomy: Another use case for #block...	['positive']
3	free coins https://t.co/DiuoePJdap	['positive']
4	RT @payvxofficial: WE are happy to announce th...	['positive']

Comparison of target labels

[3]

```
# inspect sentiment
sns.countplot(df['label'])
```

```
/opt/conda/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWa
ersion 0.12, the only valid positional argument will be `data`, and passin
n error or misinterpretation.
  FutureWarning

<AxesSubplot:xlabel='label', ylabel='count'>
```



Visualization of text length and their comparison

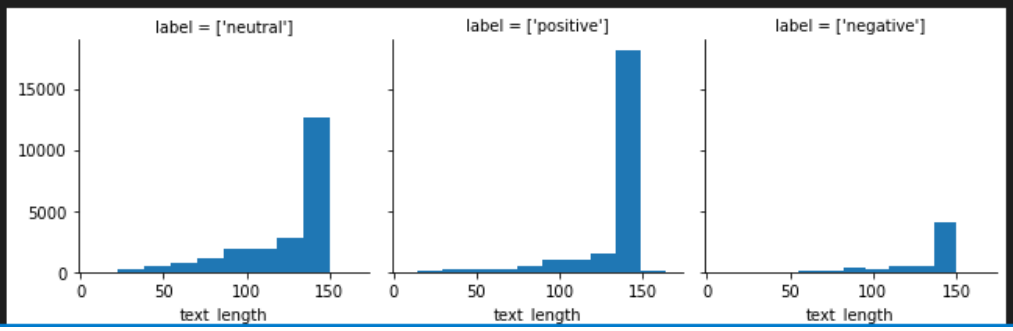
```
# text length
df['text_length'] = df['tweet'].apply(len)
df[['label','text_length','tweet']].head()
```

	label	text_length	tweet
0	['neutral']	140	RT @ALXTOKEN: Paul Krugman, Nobel Luddite. I h...
1	['neutral']	137	@lopp @_Kevin_Pham @psycho_sage @naval But @Pr...
2	['positive']	140	RT @tippereconomy: Another use case for #block...
3	['positive']	34	free coins https://t.co/DiuoePJdap
4	['positive']	146	RT @payvxofficial: WE are happy to announce th...

[5]

```
g = sns.FacetGrid(df,col='label')
g.map(plt.hist,'text_length')
```

```
<seaborn.axisgrid.FacetGrid at 0x7f0b9c6047d0>
```



Cleaning the dataset

[6]
```
def clean_text(s):
    s = re.sub(r'http\S+', '', s)
    s = re.sub('(RT|via)((?:\\b\\W*@\\w+)+)', ' ', s)
    s = re.sub(r'@\S+', '', s)
    s = re.sub('&amp', ' ', s)
    return s
df['clean_tweet'] = df['tweet'].apply(clean_text)

text = df['clean_tweet'].to_string().lower()
```

[8]
```
# Encode Categorical Variable
X = df['clean_tweet']
y = pd.get_dummies(df['label']).values
num_classes = df['label'].nunique()
```

Splitting of Train and Test sets and Tokenizing them

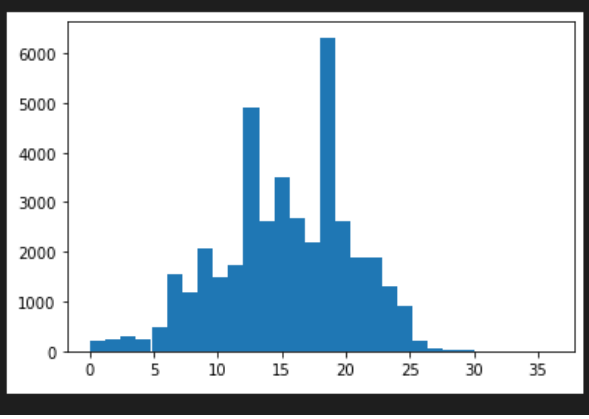
[10]
```
# Split Train Test sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.2,
                                                    stratify=y,
                                                    random_state=seed)
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

```
(40687,) (10172,) (40687, 3) (10172, 3)
```

[11]
```
# Tokenize Text
max_features = 20000
tokenizer = Tokenizer(num_words=max_features)
tokenizer.fit_on_texts(list(X_train))
X_train = tokenizer.texts_to_sequences(X_train)
X_test = tokenizer.texts_to_sequences(X_test)
```

Histogram of the tweets with respective their size

[12]
```
totalNumWords = [len(one_comment) for one_comment in X_train]
plt.hist(totalNumWords,bins = 30)
plt.show()
```



Pad-sequencing

```
[13]
max_length = 30
X_train = sequence.pad_sequences(X_train, maxlen=max_length)
X_test = sequence.pad_sequences(X_test, maxlen=max_length)
print(X_train.shape,X_test.shape)A
```

```
(40687, 30) (10172, 30)
```

```
[14]

batch_size = 128
epochs = 10
```

Building the Model

```
[16]
max_features = 20000
embed_dim = 50

np.random.seed(seed)

model = Sequential()
model.add(Embedding(max_features, embed_dim, input_length=X_train.shape[1]))

model.add(Conv1D(filters=32, kernel_size=3, padding='same', activation='relu'))
model.add(MaxPooling1D(pool_size=2))
model.add(LSTM(50, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
print(model.summary())
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 30, 50)            1000000
```

Training the model till it Overfits

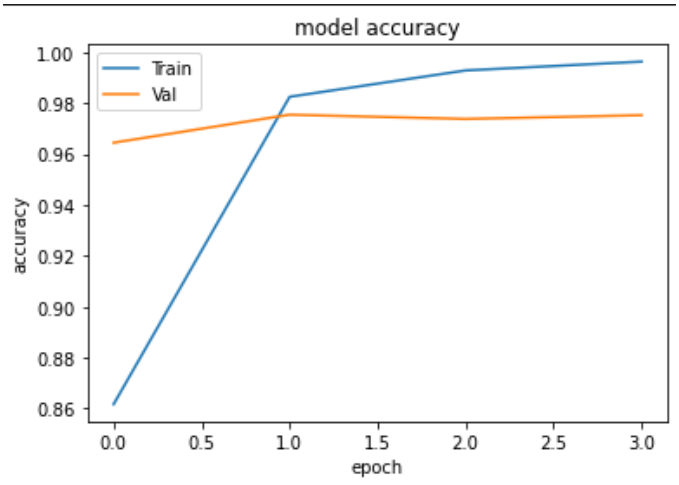
```
[17]
early_stopping = EarlyStopping(monitor='val_loss', patience=2, mode='min')

history = model.fit(X_train, y_train,
                    validation_data=(X_test, y_test),
                    epochs=epochs,
                    callbacks=early_stopping,
                    batch_size=batch_size, verbose=2)
```

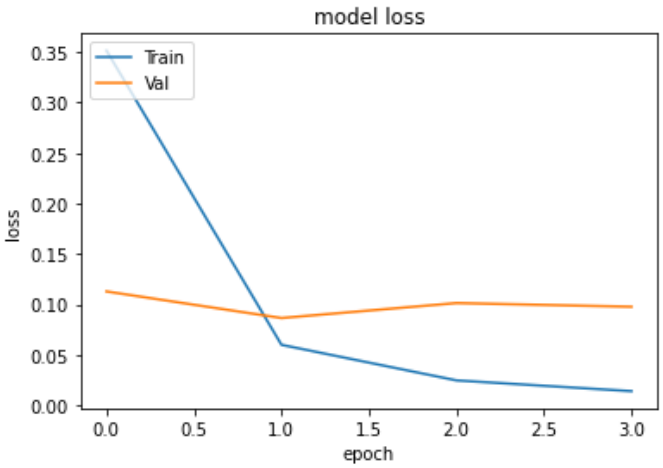
```
Epoch 1/10
318/318 - 14s - loss: 0.3517 - accuracy: 0.8617 - val_loss: 0.1128 - val_accuracy: 0.9645
Epoch 2/10
318/318 - 11s - loss: 0.0598 - accuracy: 0.9825 - val_loss: 0.0864 - val_accuracy: 0.9755
Epoch 3/10
318/318 - 11s - loss: 0.0245 - accuracy: 0.9929 - val_loss: 0.1012 - val_accuracy: 0.9738
Epoch 4/10
318/318 - 11s - loss: 0.0139 - accuracy: 0.9964 - val_loss: 0.0976 - val_accuracy: 0.9753
```

Accuracy and Loss Plots

```
[18]
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['Train', 'Val'], loc='upper left')
plt.show()
```

```
[19]
#Loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['Train', 'Val'], loc = 'upper left')
plt.show()
```





Prediction on the Test data

```
[20]
# predict class with test set
y_pred_test =  np.argmax(model.predict(X_test), axis=1)
print('Accuracy:\t{:0.1f}%'.format(accuracy_score(np.argmax(y_test,axis=1),y_pred_test)*100))
plt.tight_layout()
```

```
Accuracy:       97.5%

<Figure size 432x288 with 0 Axes>
```

Confusion Matrix and Classification report

```
[21]
#confusion matrix
confmat = confusion_matrix(np.argmax(y_test,axis=1), y_pred_test)
confmat
```

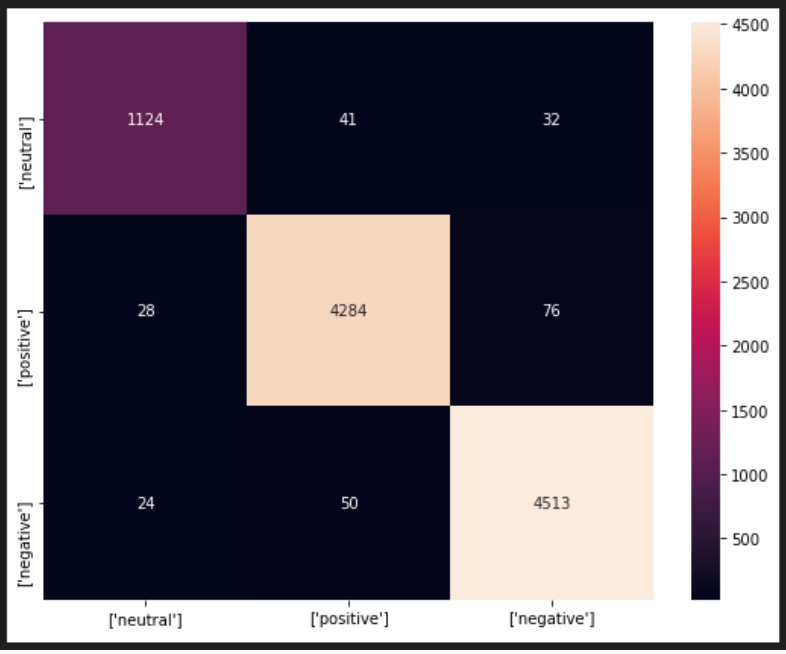
```
array([[1124,   41,   32],
       [  28, 4284,   76],
       [  24,   50, 4513]])
```

```
[22]
#classification report
print('\n')
print(classification_report(np.argmax(y_test,axis=1), y_pred_test))
```

```
              precision    recall  f1-score   support

           0       0.96      0.94      0.95      1197
           1       0.98      0.98      0.98      4388
           2       0.98      0.98      0.98      4587

    accuracy                           0.98     10172
   macro avg       0.97      0.97      0.97     10172
weighted avg       0.98      0.98      0.98     10172
```

Heatmap

```
[32]
cm = pd.DataFrame(confmat , index = [i for i in df['label'].unique()],
                  columns = [i for i in df['label'].unique()])

plt.figure(figsize = ( 9 ,7))
sns.heatmap(cm,annot=True , fmt="d")
plt.show()
```



Accuracy Results

Training	99.6%
Testing	97.5%
Validation	97.5%

Conclusion

We have used LSTM to perform sentiment analysis on the twitter dataset about bitcoins. The results show that the model has been successful and gave us good accuracies for both training and testing.