[Kaur, 5(7): July 2018]
DOI: 10.5281/zenodo.1316903

ISSN 2348 – 8034
Impact Factor- 4.022

GLOBAL JOURNAL OF ENGINEERING SCIENCE AND RESEARCHES

SENTIMENT ANALYSIS BASED ON DEEP LEARNING

Jaspreet Kaur[*1] & Er. Brahmaleen Kaur Sidhu[2]

[*1]Student: Computer Engineering, Punjabi University, Patiala, Punjab

[2]Assistant Professor: Computer Engineering, Punjabi University, Patiala, Punjab

ABSTRACT

Many emerging social sites, famous forums, review sites, and many bloggers generate huge amount of data in the form of user sentimental reviews, emotions, opinions, arguments, viewpoints etc. about different social events, products, brands, and politics, movies etc. Sentiments expressed by the users has great effect on readers, political images, online vendors. So the data present in scattered and unstructured manner needs to be managed properly and in this context sentiment analysis has got attention at very large level. Sentiment analysis can be defined as organization of the text which is used to understand the mindsets or feelings expressed in the form of different manners such as negative, positive, neutral, not satisfactory etc. This paper explains the implementation and accuracy of sentiment analysis using Tensor flow and python with any kind of text data. It works on embedding, LSTM and Sigmoid layers and finds the accuracy of data in iterative manner for better result

Keywords: *RNN, Tensor flow, Deep Learning, Sentiment Analysis, LSTM, Sigmoid*.

I. INTRODUCTION

World has seen some major advancements in computing technologies in the last decade. Natural Language understanding has become a prime important in last few years with the arrival of machine learning (ML) and advancements like deep learning [1] and artificial intelligence. Computing machines can always be programmed and made to learn things, which help them to perform some particular operations. The main problem is the inability to understand and comprehend things just like humans. But with the popularity of neural networks[2][3] and advances that it has made, deep learning[3][4] and artificial intelligence have evolved rapidly and is making revolutionary shifts in almost all the industries and it also has started to engineer machines into learning, understanding and implementing actions on their own.

Some of the examples which can be seen in the real world can be like Computers playing and beating players in games like Chess and Go, Self-Driving Cars etc. Natural Language Processing (NLP) can be used for wide range of applications which uses natural language understanding to find the meaning and context behind the text and use it to solve various problems. Text Semantics is used to understand the meaning of text or language and when words are combined to make sentences, these words then have lexical and contextual relations between them, which further leads to other various types of relationships and hierarchies. Semantics is the center point of all this to analyze the relationships and extract meaningful data from them. Semantics is totally related with the context and meaning, while the structure of the text holds little or no significance here. But there may be times when the syntax and arrangement of words helps us in extracting the meaning data from the context of words and helps us in differentiation of things like “Apple can be a fruit” from “Apple is a technology company”.

II. SENTIMENT ANALYSIS

Sentiment analysis is one of the most popular application of text-based analytics and is used in large number of web sites, mobile applications, tutorials etc. applications which focuses on analyzing sentiments from different text resources ranging from corporate surveys like Google Opinions to Movie Reviews like on International Movie Database(IMDB). The major aspect of sentiment analysis is to analyze the body of text to find and understand the opinion expressed on the basis of different factors like mood or modality etc. Sentiment analysis works best on subjective context rather than objective context. It is because when text is based on objective context, then text has

[Kaur, 5(7): July 2018] ISSN 2348 – 8034
DOI: 10.5281/zenodo.1316903 Impact Factor- 4.022

mainly comprised of normal sentences without expressing any sort of mood, emotions or feelings. Sentiment analysis is used in large extent these days in social media[5] giants like Facebook and Twitter[5], ecommerce web sites, product related web sites, movie review sites etc. by understanding the opinions of people. Textual Data, although is unstructured, mainly comprised of two types: Objective, which is also known as factual based and Subjective, which is also known as Opinion based. Social Media, Survey Sites like Google Opinion, and feedback data like Movie reviews or Product reviews etc. are mainly opinion related data which expresses the judgement, emotions and feelings of human beings.

Sentiment analysis, also popularly known as Opinion Mining can be explained as the process that using NLP techniques, lexical resources, linguistics, and machine learning (ML) to fetch data like emotions, mood and modality etc. and then use the fetched data to compute the polarity of the text document [6]. From polarity, means to find if the text document expresses positive, negative, or neutral sentiment. There can be much more advanced analysis where more complex emotions can be used like sadness, sarcasm, anger etc.

III. IMPLEMENTATION AND RESULTS

Recurrent neural networks (RNN) are connectionist models with the ability to selectively pass information across sequence steps, while processing sequential data one at a time [12]

RNN is always more accurate than using a feed forward network as it also includes the information about the sequence of words. We have used a data set of twitter posts and comments, which are accompanied by labels. Words can be passed in the embedded layer as thousands of words needs to have much more efficient input representation than using one-hot encoded vectors and then from embedded layer representations, it can be passed to the LSTM cells, which will further add recurrent network connections to help add sequence of words in the data. And in the last, LSTM cells will move to sigmoid layer which brings the output. We are using sigmoid as we are predicting of the text is having positive or negative sentiment.

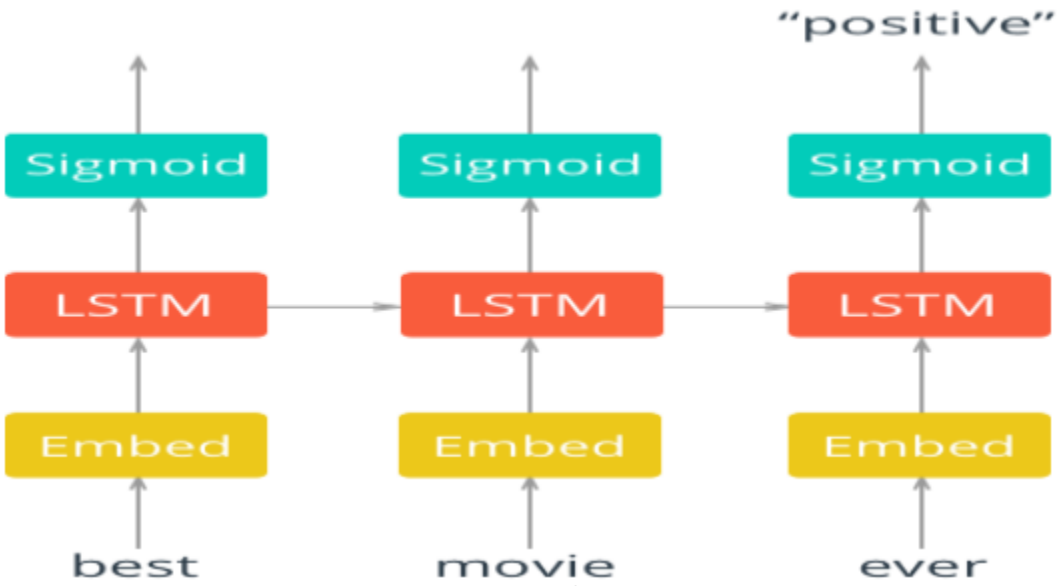

Figure 1: LSTM Model for Sentiment Analysis

Two libraries that we need to import before starting the code are:

- Numpy – import numpy as np
- Tensorflow – import tensorflow are tf

ISSN 2348 – 8034
Impact Factor- 4.022

We need to open or call the reviews and label files and read them. After reading all the content, 2000 characters are pushed from the reviews file and they will be preprocessed in the next step. When we are building a neural network, the very first step after after fetching the data from the file is to get all the data collected from the file to be in proper form so that it can be feeded in to the network . As we are using embedded layer, therefore we need to encode every word with an integer. For example, in our file of reviews we can eliminate the period operators and it has to be delimited with the new line characters i.e. \n. For that, what we will be doing is to split the text data into every review using \n as a delimiter. After that one can combine all the split reviews together into a single big string.

```
from string import punctuation
all_text = ''.join([c for c in reviews if c not in punctuation])
reviews = all_text.split('\n')

all_text = ' '.join(reviews)
words = all_text.split()
```

Figure 2: To remove punctioan

Punctuation can be removed using the above code. Then one can get all the text without the newlines and it has to be split into individual words**.**

```
words[:100]
['bromwell',
 'high',
 'is',
 'a',
 'cartoon',
 'comedy',
 'it',
 'ran',
 'at',
 'the',
 'same',
 'time',
 'as',
 'some',
 'other',
 'programs',
 'about',
 'school',
 'life',
 'such',
 'as',
 'teachers',
 'my',
 'years',
 'in',
 'the',
 'teaching',
 'profession',
```

Figure 3: Splitting into individual words

As shown above in the figure, individual words are taken out of the reviews file and next step is to encode the words which can be done by first do the embedding lookup which needs integers to be passed in to the network. The best which can be used for this is to create a dictionary that maps the words in the vocabulary and changes into integers. After that every review has to be changed into the integer so that they can also be passed to the network. Code which has been used to convert the words into integers is given below:

[Kaur, 5(7): July 2018]
DOI: 10.5281/zenodo.1316903

ISSN 2348 – 8034
Impact Factor- 4.022

```
from collections import Counter
counts = Counter(words)
vocab = sorted(counts, key=counts.get, reverse=True)
vocab_to_int = {word: ii for ii, word in enumerate(vocab, 1)}

reviews_ints = []
for each in reviews:
    reviews_ints.append([vocab_to_int[word] for word in each.split()])
```

Figure 4: To change words into integers

Above code changes words into integers. We have built a dictionary that maps the words in to integers .We will also be going to pad out input vectors with 0s, therefore it is necessity that out integers starts with 1 and not 0. The converted values from reviews to integers will be stored in a new list known as review_ints. We have used two different labels, i.e. Positive and Negative, so in order to use them, we need to convert them into 0 and 1, which is done in the below code:

```
labels = labels.split('\n')
labels = np.array([1 if each == 'positive' else 0 for each in labels])
```

Figure 5: Assignment of labels- Positive and Negative

Now after splitting the reviews, one needs to check the reviews length if there are any zero length reviews present. If there exists some zero length reviews, then they have to be discarded from the next step. So we have deleted the review which has 0 length from the reviews_ints list by using below given code:

reviews_ints = [each for each in reviews_ints if len(each) > 0]

In the next step, we have created an array with the name features and it holds the data that we have to pass on to the network. This data comes from the review_ints, as we have to feed integers to our network. Every row will be 200 elements long and padded with 0s in case words are shorter than 200 words. For example if someone's review or tweet is ['world' 'is' 'great'],[127, 58, 178] as integers, then the row will look like [0,0,0,0,……..,0,0, 127, 58, 178]. In case the review is longer than 200 words, then the first 200 words can act as feature vector. There are various methods with which we can do this, we have used the following code shown in below figure inside a Python Jupyter Notebook that also includes the output:

[Kaur, 5(7): July 2018]
DOI: 10.5281/zenodo.1316903

ISSN 2348 – 8034
Impact Factor- 4.022

```
In [11]: seq_len = 200
         features = np.zeros((len(reviews), seq_len), dtype=int)
         for i, row in enumerate(reviews_ints):
             features[i, -len(row):] = np.array(row)[:seq_len]

In [12]: features[:10,:100]

Out[12]: array([[    0,     0,     0,     0,     0,     0,     0,     0,     0,
                     0,     0,     0,     0,     0,     0,     0,     0,     0,
                     0,     0,     0,     0,     0,     0,     0,     0,     0,
                     0,     0,     0,     0,     0,     0,     0,     0,     0,
                     0,     0,     0,     0,     0,     0,     0,     0,     0,
                     0,     0,     0,     0,     0,     0,     0,     0,     0,
                     0,     0,     0,     0,     0,     0, 21025,   308,     6,
                     3,  1050,   207,     8,  2138,    32,     1,   171,    57,
                    15,    49,    81,  5785,    44,   382,   110,   140,    15,
                  5194,    60,   154,     9,     1,  4975,  5852,   475,    71,
                     5,   260,    12, 21025,   308,    13,  1978,     6,    74,
                  2395],
                [    0,     0,     0,     0,     0,     0,     0,     0,     0,
                     0,     0,     0,     0,     0,     0,     0,     0,     0,
                     0,     0,     0,     0,     0,     0,     0,     0,     0,
                     0,     0,     0,     0,     0,     0,     0,     0,     0,
                     0,     0,     0,     0,     0,     0,     0,     0,     0,
                     0,     0,     0,     0,     0,     0,     0,     0,     0,
                     0,     0,     0,     0,     0,     0,     0,     0,     0,
```

Figure 6: Converting into matrix

After getting all the data in proper form in an array of integers, Training and validation will be done by creating a training and validation set. We have created sets for features and the labels. We have created a split fraction variable i.e. split_frac, which is the fraction of the data that is to be kept in the training dataset with the code set given below:

```
In [13]: split_frac = 0.8
         split_idx = int(len(features)*0.8)
         train_x, val_x = features[:split_idx], features[split_idx:]
         train_y, val_y = labels[:split_idx], labels[split_idx:]

         test_idx = int(len(val_x)*0.5)
         val_x, test_x = val_x[:test_idx], val_x[test_idx:]
         val_y, test_y = val_y[:test_idx], val_y[test_idx:]

         print("\t\t\tFeature Shapes:")
         print("Train set: \t\t{}".format(train_x.shape),
               "\nValidation set: \t{}".format(val_x.shape),
               "\nTest set: \t\t{}".format(test_x.shape))

                          Feature Shapes:
Train set:                (20000, 200)
Validation set:           (2500, 200)
Test set:                 (2501, 200)
```

Figure 7: Training and Validation

As we have 25000 line and labels dataset, we have splitted it with the train and text fractions of 0.8, 0.1 and 0.1 with final shapes look like in the following format:

[Kaur, 5(7): July 2018]
DOI: 10.5281/zenodo.1316903
ISSN 2348 – 8034
Impact Factor- 4.022

Feature Shapes:
Train set: (20000, 200)
Validation set: (2500, 200)
Test set: (2501, 200)

Build the graph:
We have defined various parameters in this to create a graph:
Lstm_size – It is the no. of units in the LSTM's hidden layers. If larger, provides great performance. Mostly used values are 128, 256, 512, 1024 etc.
Lstml_layers - It is the no. of LSTM layers in the network. We have started it with one.
Batch_size – It means the no. of reviews which can be pushed in a single training pass.
Learning_rate
We are using 200 element large review vectors. Every batch consists of batch_size vectors. As we are using tensorflow library, we have used some inbuilt functions of tensorflow to create some placeholders i.e. inputs, labels and keep_prob by using tf.placeholder. labels. Keep_prob is a 0-D tensor, therefore there is no need to provide the size to tf.placeholder. Below is the code used to use tensorflow to create 0-D and 2-D tensors.

```
lstm_size = 256
lstm_layers = 1
batch_size = 500
learning_rate = 0.001
```

```
n_words = len(vocab)

# Create the graph object
graph = tf.Graph()
# Add nodes to the graph
with graph.as_default():
    inputs_ = tf.placeholder(tf.int32, [None, None], name='inputs')
    labels_ = tf.placeholder(tf.int32, [None, None], name='labels')
    keep_prob = tf.placeholder(tf.float32, name='keep_prob')
```

Figure 8: To use Tensor Flow to create 0-D and 2-D tensors

Embedding: Embedding is a necessary part of our work as number of words in our vocabulary is over 70,000 and it will create a mess if we do one-hot-encode for our classes. What we have done is that we have creates an embedding layer and then we used that layer as a lookup table. We have created a new layer and make the network learn about the weights. We have created an embedded lookup matrix using a tf.Variable. Then, we have used that embedded matrix to find the embedded vectors which can be passed to the LSTM cells using tf.nn.embedding_lookup. What this function does is that it will take embedding matric and a review vector as an input tensor and updates or output with another tensor with the embedded vectors. Therefore in case the embedding layer consists of 300 units, then the function will return the tensor with size of [batch_size , 300]

```
embed_size = 300

with graph.as_default():
    embedding = tf.Variable(tf.random_uniform((n_words, embed_size), -1, 1))
    embed = tf.nn.embedding_lookup(embedding, inputs_)
```

Figure 9: Embedding

[Kaur, 5(7): July 2018] ISSN 2348 – 8034
DOI: 10.5281/zenodo.1316903 Impact Factor- 4.022

After Embedding, now we will create LSTM cells which will be in use in recurrent networks with Tensorflow and we will define how the cell will show up. It is not like we are building a graph, but it actually is defining the type of cells in a graph. We have created RNNs using the below code:

```
with graph.as_default():
    # Your basic LSTM cell
    lstm = tf.contrib.rnn.BasicLSTMCell(lstm_size)

    # Add dropout to the cell
    drop = tf.contrib.rnn.DropoutWrapper(lstm, output_keep_prob=keep_prob)

    # Stack up multiple LSTM layers, for deep learning
    cell = tf.contrib.rnn.MultiRNNCell([drop] * lstm_layers)

    # Getting an initial state of all zeros
    initial_state = cell.zero_state(batch_size, tf.float32)
```

Figure 10: Creation of LSTM cell

Above figure shows the code that is to be used to create the LSTM cell. We have also added drop out to it with the help of Dropout Wrapper. Then we created multiple LSTM layers using MultiRNNCell. Now in the last phase, we need to run the data in to the RNN nodes. We have used tensorflow's dynamic_rnn function to achieve this:

```
with graph.as_default():
    predictions = tf.contrib.layers.fully_connected(outputs[:, -1], 1, activation_fn=tf.sigmoid)
    cost = tf.losses.mean_squared_error(labels_, predictions)

    optimizer = tf.train.AdamOptimizer(learning_rate).minimize(cost)
```

Figure 11: Creation of multiple LSTM layers using MultiRNNCell

To check the accuracy, we have used the following code, all the data goes inside the checkpoint directory and it is the prerequisite that checkpoint directory has to be there in the Pycharm directory of Jupyter notebook, if one is using Pycharm synced with python interpreter and Jupyter notebook, if checkpoints directory is not present, then our code will not work and brings an error in case of our code:

[Kaur, 5(7): July 2018]
DOI: 10.5281/zenodo.1316903
ISSN 2348 – 8034
Impact Factor- 4.022

```
epochs = 10

with graph.as_default():
    saver = tf.train.Saver()

with tf.Session(graph=graph) as sess:
    sess.run(tf.global_variables_initializer())
    iteration = 1
    for e in range(epochs):
        state = sess.run(initial_state)

        for ii, (x, y) in enumerate(get_batches(train_x, train_y, batch_size), 1):
            feed = {inputs_: x,
                    labels_: y[:, None],
                    keep_prob: 0.5,
                    initial_state: state}
            loss, state, _ = sess.run([cost, final_state, optimizer], feed_dict=feed)

            if iteration%5==0:
                print("Epoch: {}/{}".format(e, epochs),
                      "Iteration: {}".format(iteration),
                      "Train loss: {:.3f}".format(loss))

            if iteration%25==0:
                val_acc = []
                val_state = sess.run(cell.zero_state(batch_size, tf.float32))
                for x, y in get_batches(val_x, val_y, batch_size):
                    feed = {inputs_: x,
                            labels_: y[:, None],
                            keep_prob: 1,
                            initial_state: val_state}
                    batch_acc, val_state = sess.run([accuracy, final_state], feed_dict=feed)
                    val_acc.append(batch_acc)
                print("Val acc: {:.3f}".format(np.mean(val_acc)))
            iteration +=1
    saver.save(sess, "checkpoints/sentimenttwitter1.ckpt")
```

Figure 13: To run all iterations and Epoch

Above code uses tensorflow session function and run all the iterations and Epoch to provide the accuracy in accuracy level.

Below is the iterative output of RNN based sentiment analysis accuracy:

[Kaur, 5(7): July 2018]
DOI: 10.5281/zenodo.1316903

ISSN 2348 – 8034
Impact Factor- 4.022

```
Epoch: 3/10 Iteration: 130 Train loss: 0.141
Epoch: 3/10 Iteration: 135 Train loss: 0.128
Epoch: 3/10 Iteration: 140 Train loss: 0.104
Epoch: 3/10 Iteration: 145 Train loss: 0.138
Epoch: 3/10 Iteration: 150 Train loss: 0.119
Val acc: 0.809
Epoch: 3/10 Iteration: 155 Train loss: 0.095
Epoch: 3/10 Iteration: 160 Train loss: 0.104
Epoch: 4/10 Iteration: 165 Train loss: 0.087
Epoch: 4/10 Iteration: 170 Train loss: 0.097
Epoch: 4/10 Iteration: 175 Train loss: 0.091
Val acc: 0.786
Epoch: 4/10 Iteration: 180 Train loss: 0.112
Epoch: 4/10 Iteration: 185 Train loss: 0.105
Epoch: 4/10 Iteration: 190 Train loss: 0.156
Epoch: 4/10 Iteration: 195 Train loss: 0.142
Epoch: 4/10 Iteration: 200 Train loss: 0.122
Val acc: 0.770
Epoch: 5/10 Iteration: 205 Train loss: 0.094
Epoch: 5/10 Iteration: 210 Train loss: 0.099
Epoch: 5/10 Iteration: 215 Train loss: 0.105
Epoch: 5/10 Iteration: 220 Train loss: 0.098
Epoch: 5/10 Iteration: 225 Train loss: 0.084
Val acc: 0.826
Epoch: 5/10 Iteration: 230 Train loss: 0.092
Epoch: 5/10 Iteration: 235 Train loss: 0.075
Epoch: 5/10 Iteration: 240 Train loss: 0.076
Epoch: 6/10 Iteration: 245 Train loss: 0.078
Epoch: 6/10 Iteration: 250 Train loss: 0.115
Val acc: 0.808
```

Figure 14: Accuracy after each 5 iterations

As you can see from the above output, we did iterative accuracy to split it into smaller parts for better result and we have an average accuracy of 0.8. As outputs are taken on iterative basis, therefore below graph is used to plot the accuracy at each epoch interval:

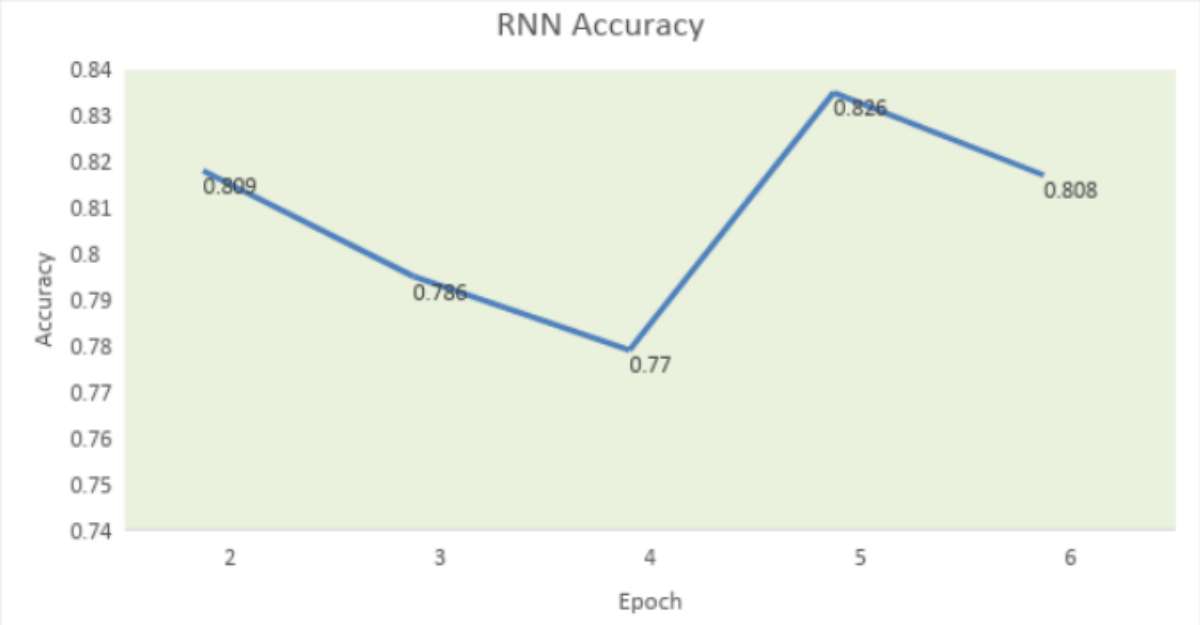

Figure 15: Graph to show RNN Accuracy

[Kaur, 5(7): July 2018] ISSN 2348 – 8034
DOI: 10.5281/zenodo.1316903 Impact Factor- 4.022

IV. CONCLUSION

Sentiment Analysis is the application which is used by many businesses to expand their growth. It uses machine learning and deep learning algorithms which are integrated with the text mining to bring meaningful data from unstructured data. This meaningful data brings lots of benefits to customers and businesses to get a better insight of the products and emotions and mood of the people on some particular topic like Elections or Movies. RNN is one of the most used deep learning techniques to find the sentiment and accuracy of the text. RNN is particularly used in case of large datasets. Large Datasets are pushed to the embedding layer by splitting them which then further are pushed to LSTM cells and at last forwarded to sigmoid layer. Python Jupyter Notebook is used in our testing and we split the output using multiple iterations defined under epoch levels for better view of the accuracy at different intervals and found accuracy to be around 0.8 which is good as we have built a system where any type of text related data can be pushed to get the sentiments and its accuracy.

REFERENCES

1. *Oscar Araque ,Ignacio Corcuera-Platas, J.Fernando Sánchez-Rada, CarlosA. Iglesias(2017),” Enhancing deep learning sentiment analysis with ensemble techniques in social applications”, . Retreived from-http://dx.doi.org/10.1016/j.eswa.2017.02.*
2. *Ming Jiang, Liqiang Jin, Feiwei Qin, Min Zhang, Ziyang Li(2016),” Network Public Comments Sentiment Analysis Based on Multilayer Convolutional Neural Network”, IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData.*
3. *Aliaksei Severyn abd Aliaksei Severyn(2015),” Twitter Sentiment Analysis withDeep Convolutional Neural Networks”, ACM. ISBN 978-1-4503-3621-5/15/08.*
4. *Adyan Marendra Ramadhani and Hong Soon Goo(2017),” Twitter Sentiment Analysis using Deep Learning Methods”, International Annual Engineering Seminar (InAES).*
5. *Kiruthika M. , Sanjana Woona and Priyanka Giri(2016),”Sentiment Analysis of Twitter Data”,International Journal of Innovations in Engineering and Technology(IJIET).*
6. *Metin Bilgin and Izzet Fatih Senturk(2017),” Sentiment Analysis on Twitter data with Semi-Supervised Doc2Vec”, UBMK International conference on computer science and engineering.*
7. *Nehal Mamgain, Ekta Mehta, Ankush Mittal and Gaurav Bhatt(2016),” Sentiment Analysis of Top Colleges in India Using Twitter Data”, International Conference on Computational Techniques in Information and Communication Technologies (ICCTICT).*
8. *J˙onatas Wehrmann, Willian Becker, Henry E. L. Cagnini, and Rodrigo C. Barros(2017),” A Character-based Convolutional Neural Network for Language-Agnostic Twitter Sentiment Analysis”,IEEE.*
9. *Vishal A. Kharde and S.S. Sonawane(2016),” Sentiment Analysis of Twitter Data: A Survey of Techniques”, International Journal of Computer Applications (0975 – 8887) Volume 139 – No.11.*
10. *J. Islam and Y. Zhang, Visual Sentiment Analysis for Social Images Using Transfer Learning Approach, 2016 IEEE Int. Conf. Big Data Cloud Comput. (BDCloud), Soc. Comput. Netw. (SocialCom), Sustain. Comput. Commun., pp. 124130, 2016.*
11. *A. Severyn and A. Moschitti, Twitter Sentiment Analysis with Deep Convolutional Neural Networks, Proc. 38th Int. ACM SIGIR Conf. Res. Dev. Inf. Retr. - SIGIR 15, pp. 959962, 2015.*
12. *Zachary C. Lipton, John Berkowitz, Charles Elkan, A Critical Review of Recurrent Neural Networks*
13. *for Sequence Learning arXiv:1506.00019v4 [cs.LG] 17 Oct 2015*
14. *H. Yanagimoto, M. Shimada, and A. Yoshimura, Document similarity estimation for sentiment analysis using neural network, 2013 IEEE/ACIS 12th Int. Conf. Comput. Inf. Sci., pp. 105110, 2013.*
15. *C. Z. Jiajun Zhang, Neural Networks in Machine Translation: An Overview, IEEE Intell. Syst., pp. 17241734, 2015*