Sarcasm Sentiment Discernment towards Deep Learning

Vijay Kumar Sharma
Department of Computer Science and Engineering
MIET, Meerut, India

Swati Sharma
Department of Information Technology
MIET, Meerut, India

Abstract- Nowadays, the use of social media is increasing with a tremendous rate and thus there has been a lot of opinionated textual data available over the internet. Sentiment Analysis is preferred for the analysis of opinionated text. There are numerous challenges to analyze the opinionated text and Sarcasm detection is one of the biggest challenges to handle. Working on sarcasm is an exhilarating summon in the field of Sentiment Analysis (SA). Sarcasm is something antithetical what an individual says; intimating frowzy feelings in conjunction with positive words. In this paper, we are developing an automated sarcasm detection model using Machine Learning and Deep Learning. Moreover, for handling of data loss while training and testing, padded sequence is inherited. During sentiment analysis, sarcasm detection is crucial. A model has been developed using Tkinter which tells whether the entered text is sarcastic or not. The accuracy of the system is tested as 99%. The focal point of this work is its accuracy.

Keywords – Tkinker, Deep Learning, Machine Learning, Sarcasm, Tokenization

I. INTRODUCTION

Sarcasm is combination of negative feeling and positive words. It is something contradiction of one's mind and ones' tongue. Sarcasm can be taken in both forms; as fun or as criticism. An analyst has to examine the intention of the user as well the context in which it is stated. It is more ubiquitous with emoticons, exclamation marks, capital letters or slang [1]. In this paper, we have developed a model to detect whether a sentence is sarcastic or not.

There are numerous classes of sarcasm [2], discussed below:

Class 1:
Attribute: Conjunction of negative and positive [-1, +1]
Example: It is not honey, it is called exquisite honey.

Class 2:
Attribute: Negative sentence followed by positive sentence or vice-a-versa
Example: Such sugary mangoes. They are spoiled in packet.

Class 3:
Attribute: Dilemma in sentence.
Example: Thinking again and again, whether to buy shorts as I am getting fat.

Class 4:
Attribute: Negative phrase followed by positive phrase or vice-a-versa
Example: Packing was good but not the product.

Class 5:
Attribute: Contrast among bad and worse.

Example: Shit, I ordered the food but it's good that I ordered only one wish from this restaurant.

Class 6:
Attribute: Contrasting with better.
Example: Amazon is giving more discount than Myntra.

Class 7:
Attribute: Indicating negativity of the targeted product.
Example: It is the least shop, I will ever visit.

Class 8:
Attribute: No distinct negative point or positive point.
Example: I can't say whether I will complete this movie.

II. RESEARCH METHODOLOGY

The proposed methodology is shown in Fig. 1. It starts with parsing the dataset, then tokenization and sequencing which includes data preprocessing i.e. making the data comfortable for the system to process. Then, is setting the model parameters, theses parameters differ from application and dataset as well. After setting the model, the model is ready for training and testing. Lastly is setting up the GUI application using Tkinter, it provides very powerful and object-oriented interface for the GUI tool kit and finally the results are displayed [3] [4].

A. Parsing the Dataset

Figure 1. shows the steps for parsing the dataset like importing the required packages get the dataset, separate the JSON into sentences and labels and splitting the dataset into training and testing. Parsing the dataset requires importing following packages:

```
>>> import json

>>> import tensorflow as tf import numpy as np import pandas as pd
```

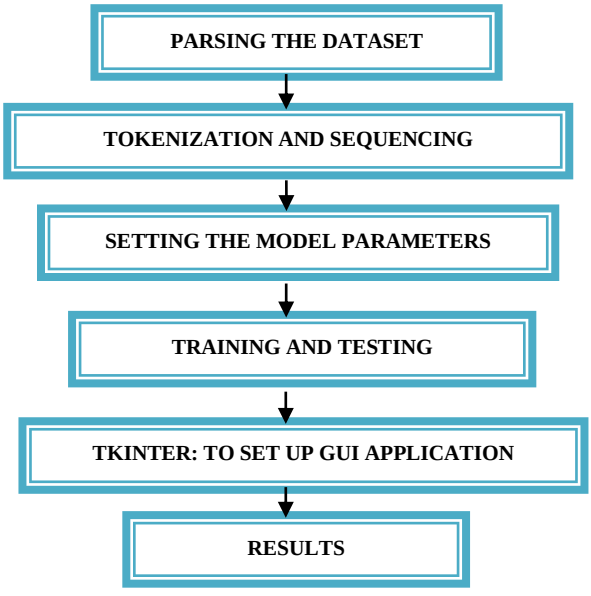

Figure 1. Steps for Sarcasm Detection Model

```
>>> from tensorflow.keras.preprocessing.text import Tokenizer
>>> from tensorflow.keras.preprocessing.sequence import pad_sequences
```

After importing the required packages, the dataset is being loaded:

```
>>> srcsm_json = pd.read_json('Sarcasm_Headlines_Dataset.json', lines = True)
```

Then, the dataset is divided into sentences and labels:

```
>>>sentences = srcsm_json[‘headline’]
>>>labels = srcsm_json[‘is_sarcastic’]
>>>for item in srcsm_json.json()
>>> sentences.append(item[‘headline’]) labels.append(item[‘is sarcastic’])
```

Further, the dataset is divided into train and test:

```
training_sentences = sentences[0:training_size]
testing_sentences = sentences[training_size:]
training_labels = labels[0:training_size]
```

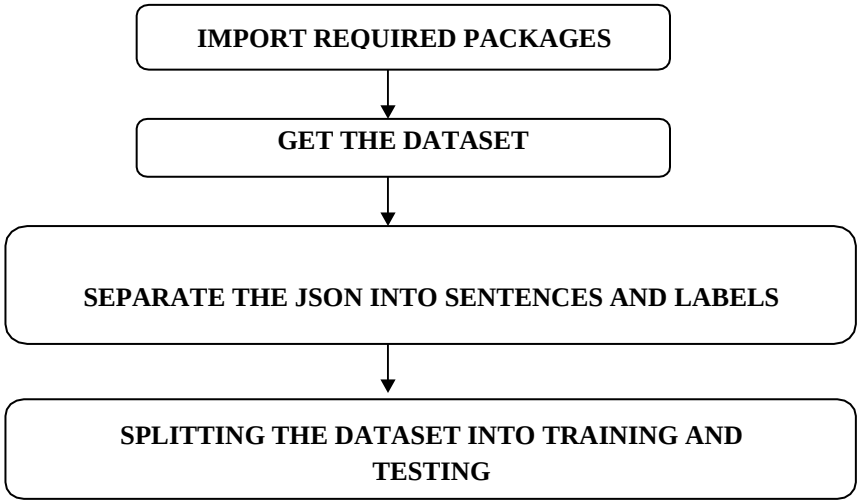

Figure 2. Parsing the dataset

The data set comprises of the following headers:

- article_link: hold link of news articles.
- headline: hold headline of news.
- is_sarcastic: hold 0 for non-sarcasm,

1 for sarcasm.

Table 1. shows that the sentence is sarcastic or not according to the used dataset.

Table 2. Sample of dictionary representing sarcasm or non-sarcasm

article link	Headline	is_sarcastic
"https://www.huffingtonpost.com/entry/elections-in-ukraine_b_5380334.html"	"elections in ukraine"	0
"https://www.huffingtonpost.com/entry/im-really-upset-about-the-midterm-elections-god-i-love-my-iphone_b_6116322.html"	"i'm really upset about the midterm elections god i love my new iphone"	0
"https://www.theonion.com/after-careful-consideration-bush-recommends-oil-drilli-1819586993"	"after careful consideration, bush recommends oil drilling"	1
"https://www.huffingtonpost.com/entry/election-2016-leaving-america_b_10377856.html"	leaving america after the elections? here's a great option.	0
"https://www.theonion.com/life-much-better-thanks-to-recent-elections-1819564960"	"headline": "life much better thanks to recent elections"	1

B. Tokenization and Sequencing

The dataset needs to be preprocessed so that the computer can process it effectively. For this, tokenization and sequencing is being used.

Tokenization divides the complete document into individual sentences or words as desired which becomes easy to read and understand the accurate meaning, referred as tokens. Figure. 3 shows the steps for tokenization and sequencing.

Setting Tokenizer properties

```
vocab_size = 10000
```

Fit the tokenizer on training data

```
tokenizer = Tokenizer(num_words=vocab_size, oov_token=oov_tok)
tokenizer.fit_on_texts(training_sentences)
word_index = tokenizer.word_index
```

Setting the padding properties

```
max_length = 100
trunc_type='post'
padding_type='post'
```

Creating padded sequence from train and test data

```
training_es = tokenizer.texts_to_sequences(training_sentences)
```

```
training_padded = pad_sequences(training_sequences, maxlen=max_length, padding=padding_type, truncating=trunc_type)
testing_sequences = tokenizer.texts_to_sequences(testing_sentences)
testing_padded = pad_sequences(testing_sequences, maxlen=max_length, padding=padding_type, truncating=trunc_type)
```

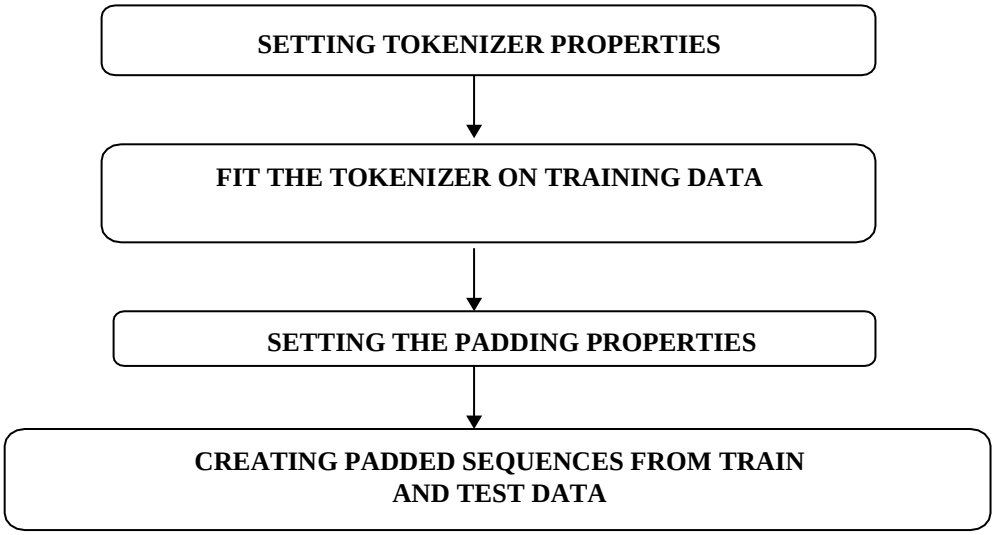

Figure 3. Tokenization and sequencing

C. Setting the Model Parameters

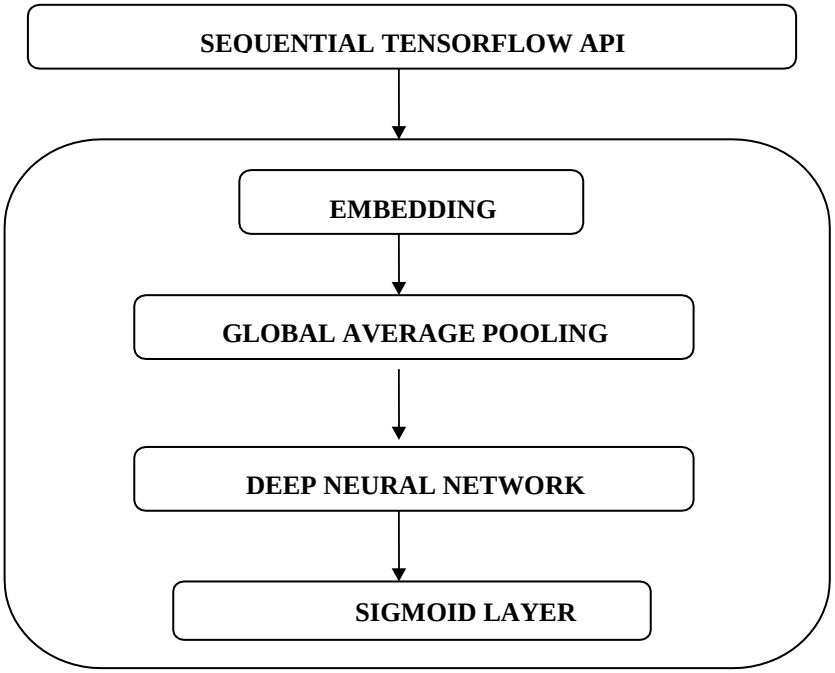

Figure 4. Setting Model Parameters

Figure. 4 shows the setting up of model parameters. TensorFlow has two API's; Sequential API and Functional API. Here, Keras Sequential API is being used. In keras Sequential model, there is a stack of layer entangled in a single pipeline. Its layers are accessed using 'layers' attribute.

The top layer is embedding which gets direction of all the words. Then, is 'Global Average Pooling' which sums all vectors. Then is 'Deep Neural Network' which validates the data from the article taken and last is 'Sigmoid Layer' which returns whether the input is sarcastic or not.

model = tf.keras.Sequential([tf.keras.layers.Embedding(vocab_size,embedding_dim, input_length=max_length), tf.keras.layers.GlobalAveragePooling1D(), tf.keras.layers.Dense (24, activation='relu'), tf.keras.layers.Dense(1, activation = 'sigmoid')])

D. Training and Testing

Figure 5 shows the steps to be followed for training and testing the developed model. It involves converting the list into NumPy arrays, after converting the model is trained and then the testing is done for checking the accuracy of the trained model.

Convert the list into NumPy:

```
training_padded = np.array(training_padded)
training_labels = np.array(training_labels)
testing_padded = np.array(testing_padded)
testing_labels = np.array(testing_labels)
```

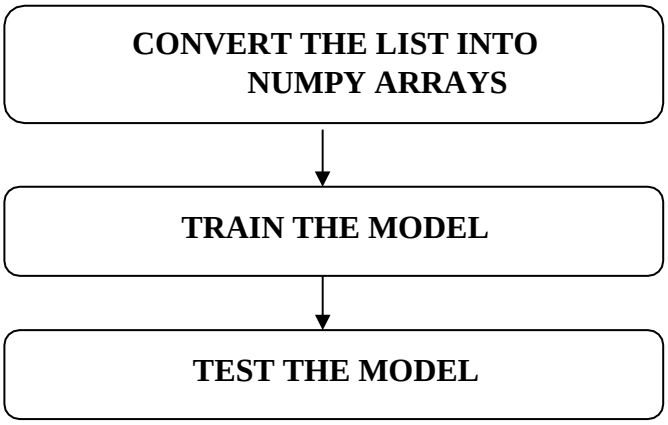

Figure 5. Training and Testing

Train and Test the Model

```
num_epochs = 30
history = model.fit(training_padded, training_labels, epochs=num_epochs,
validation_data=(testing_padded, testing_labels), verbose=2)
```

Figure 6 shows the sample of training and testing with train_loss, train_accuracy, test_loss and test_accuracy.

In total, there are 30 epochs, as in the 30th epoch the 99.39% accuracy of the developed model is achieved.

E. Setting up GUI Application using Tkinter

Tkinter, an object oriented layer over Tcl/Tk. Tk is not section of Python; it is prolonged at the active state. It is an interface for Tk GUI toolkit embedded with Python. Python when entangled with Tkinter bestow with fastest and easier method for creation of GUI application. It provides very powerful, an object-oriented interface for the Tk GUI tool kit. Figure 5.9 shows the steps for creating the GUI application. Steps for creating GUI application using Tkinter:

(i) Import Tkinter package, and handling of its errors.

```
try:
        import Tkinter as tk
except ImportError:
        import tkinter as tk
try:
        import tkk py3 = False
except ImportError: import
        tkinter.ttk as ttk py3 = True
```

(ii) Creating GUI application main window.

(iii) Adding widgets to GUI application.

(iv) Enter main-event loop for taking action for each event as triggered by user.

```
if__name__==  '__main__':  sentence,
                sarcastic = None, None
root = tk.Tk() toplevel1(root)  root.mainloop()
```

Tkinter provides 19 types of widgets, such as, Button, Canvas, Checkbutton, Entry, frame, Label, Listbox, Menubutton, Menu, Message, Radiobutton, Scale, Scrollbar, Text, Toplevel, Spinbox, PanelWindow, LabelFrame and tkMessageBox.Here, Button, Entry and Label are used.

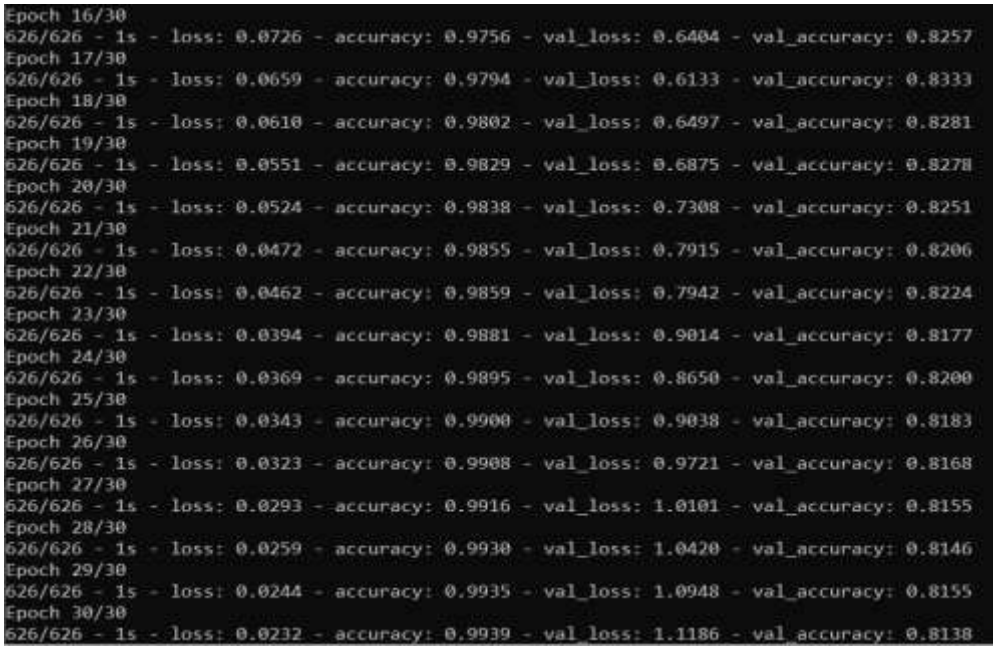

Figure 6. Glimpse of training

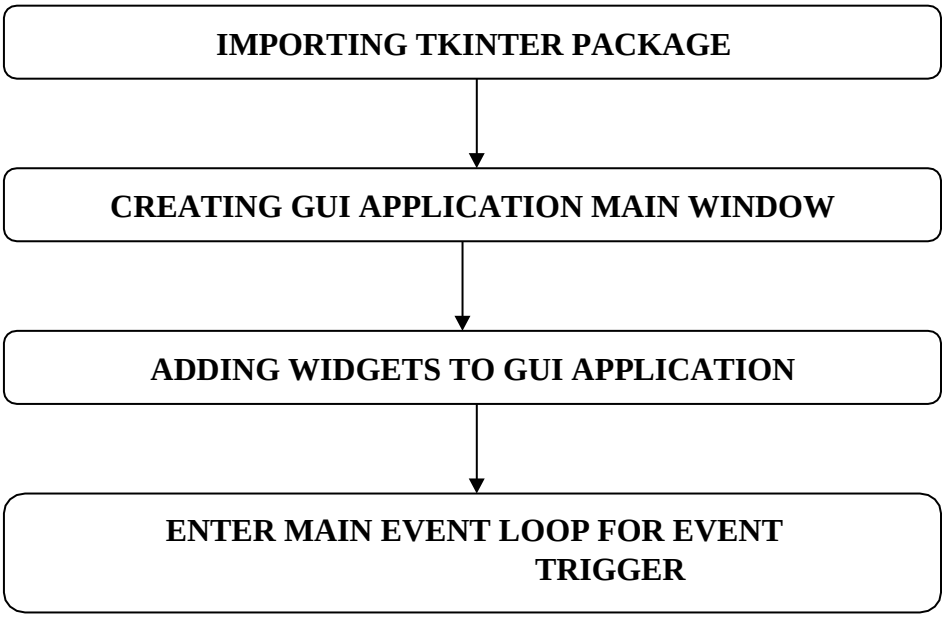

Figure 7. Steps for setting up GUI application

Figure 8. shows the screenshot of the developed model.

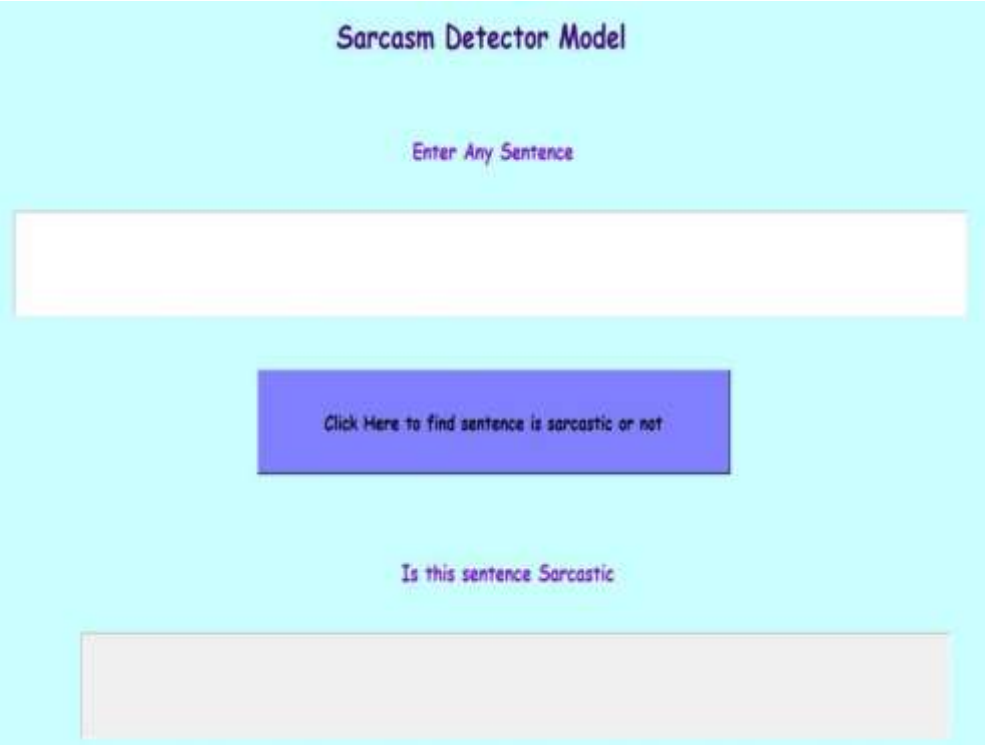

Figure 8. Screenshot of the developed model

III. RESULT AND DISCUSSION

Table 2 shows the training accuracy and testing accuracy at different epochs. In total 30 epochs, the developed model has received its highest accuracy as 99.39%.

Table 2. Training and Testing accuracy at different epochs

EPOCH	ACCURACY	TEST_ACCURACY
1	0.6030	0.8052
2	0.8406	0.8460
3	0.8818	0.8540
4	0.9019	0.8498
5	0.9182	0.8568
6	0.9280	0.8549
7	0.9385	0.8555
8	0.9441	0.8547
9	0.9522	0.8514
10	0.9580	0.8511
11	0.9616	0.8390
12	0.9642	0.8471
13	0.9686	0.8438
14	0.9719	0.8403
15	0.9742	0.8356
16	0.9756	0.8257
17	0.9794	0.8333
18	0.9802	0.8281
19	0.9829	0.8278
20	0.9838	0.8251
21	0.9855	0.8206
22	0.9859	0.8224
23	0.9881	0.8177
24	0.9895	0.8200
25	0.9900	0.8183
26	0.9908	0.8168
27	0.9916	0.8155
28	0.9930	0.8146
29	0.9935	0.8155
30	0.9939	0.8138

The graph in Figure 9. shows the training accuracy at different epochs. Figure 10 shows the testing accuracy at different epochs and Figure 11 shows the comparison among training and testing accuracy.

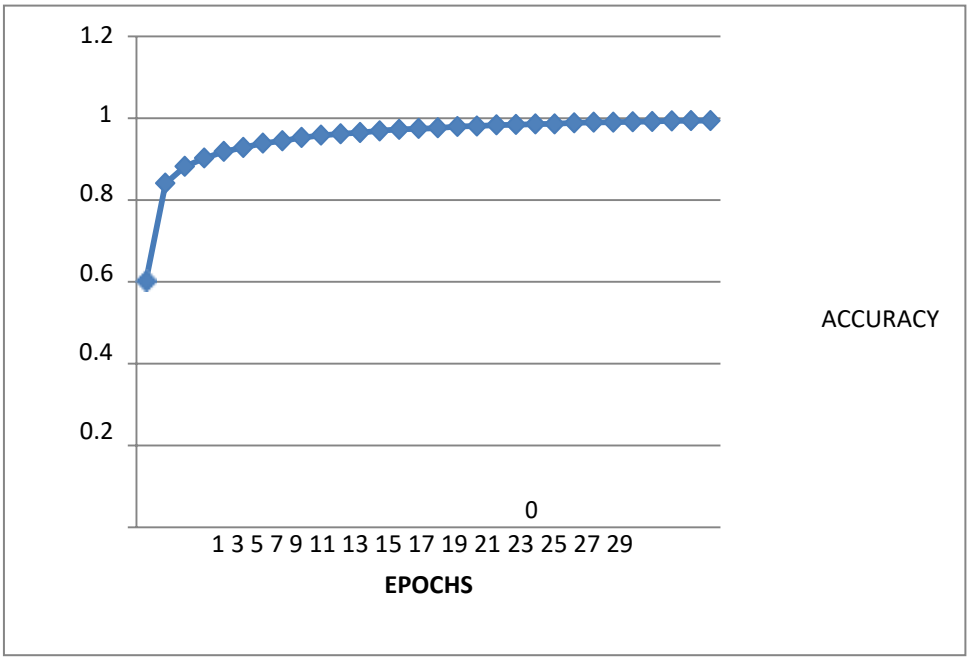

Figure 9. Training_Accuracy

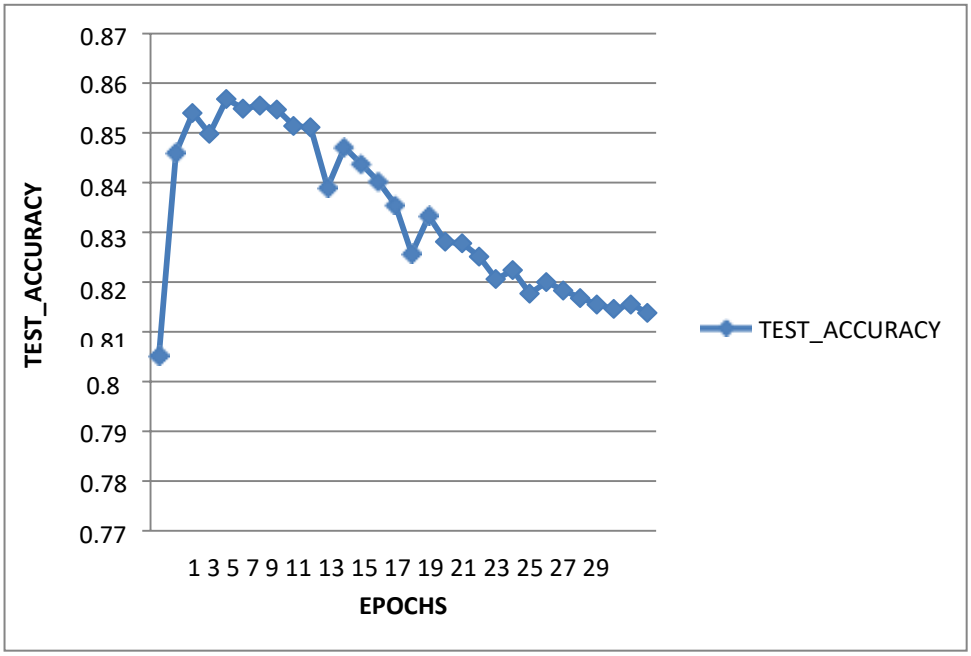

Figure 10. Testing_Accuracy

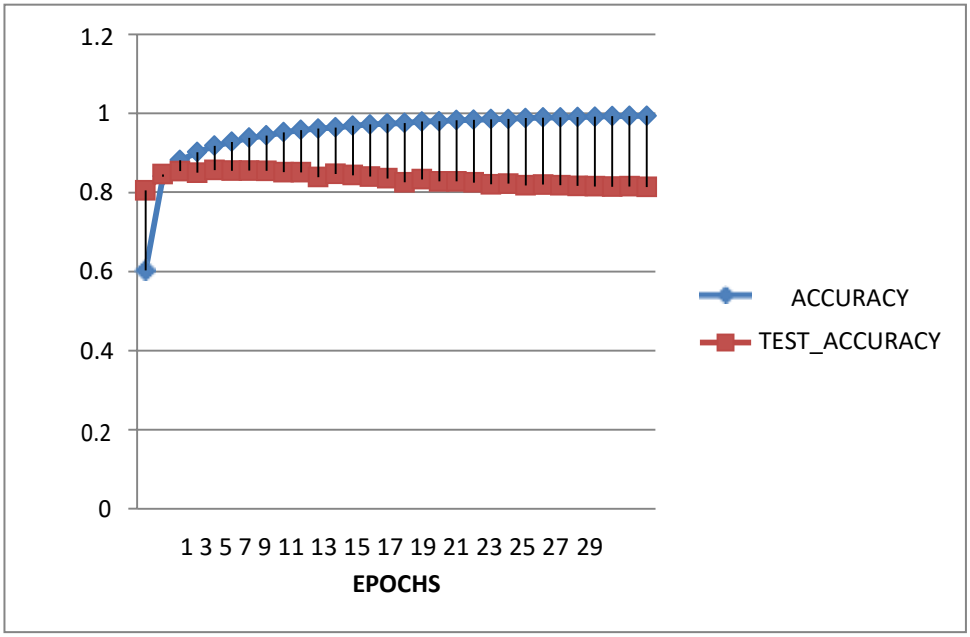

Figure 11. Comparison among Training and Testing Accuracy

The model was able to attain 99% accuracy on training data set and 81% accuracy on test data set.

IV.CONCLUSION

A model is developed for identifying whether the statement is sarcastic or not. For training, a dataset has been taken. The results showed that the developed model has achieved 99.39% accuracy in 30 epochs. The future work can be done on irony any satire as well.

REFERENCES

[1] Hussain, M.M, Howard, P,N.: The role of digital media. In: Journal of Democracy, vol. 22, pp. 35–48, (2011).

[2] Tam, N.T., Hai, D.T., Son, L.H., Vinh, L.H.: Improving lifetime and network connections of 3D wireless sensor networks based on fuzzy clustering and particle swarm optimization, Wireless Netw., vol. 24, pp. 1477–1490, no. 5, (2018).

[3] Saravanan, K., Aswini, S., Kumar, R., Son, L.H.: How to prevent maritime border collision for Fisheries?—A design of real-time automatic identification system, pp. 1–12. Earth Sci. Inform. doi: 10.1007/s12145-018-0371-5. (2019).

[4] Robinson, Y.H., Julie, E.G., Saravanan, K., Kumar, R., Son, L.H.: FDAOMDV: Fault-tolerant disjoint ad-hoc on-demand multipath distance vector routing algorithm in mobile ad-hoc networks, J. Ambient Intell. Hum. Comput., pp. 1–18. doi: 10.1007/s12652-018-1126-3, (2019).

[5] Son, L.H., Fujita, H.,: Neural-fuzzy with representative sets for prediction of student performance, Appl. Intell., vol. 49, no. 1, pp. 172–187, (2019).

[6] Kumar, A., Jaiswal, A.,: Systematic literature review of sentiment analysis on Twitter using soft computing techniques Concurrency Comput., Pract. Exper. p. e5107. doi: 10.1002/cpe.5107, , (2019).

[7] Cho, S., Cha, M., Kim, Y., Song, J., Sohn, K.,: Investigating Temporal and Spatial Trends of Brand Images Using Twitter Opinion Mining, In: Information Science and Applications International Conference, pp. 1-4, (2014).

[8] Diakopoulos, N.A., Shamma, D.A.: Characterizing debate performance via aggregated twitter sentiment, pp. 1195-1198, Proc. of the 28th Int. Conf. on Human factors in computing systems, Atlanta, Georgia, USA, (2010).

[9] Kucuktun, O., Cambazoglu, B., Weber, I., Ferhatosman, H.: A large-scale sentiment analysis for yahoo! Answers. In: Proceedings of the fifth ACM International Conference on Web search and data mining, pp. 633–642, ACM, (2012).

[10] Thanh, N.D., Ali, M., Son, L.H.: A novel clustering algorithm in a neutrosophic recommender system for medical diagnosis, Cogn. Comput., vol. 9, no. 4, pp. 526–544 (2017).

[11] Thong, P.H., Son, L.H.: Picture fuzzy clustering: A new computational intelligence method, Soft Comput., vol. 20, no. 9, pp. 3549–3562 (2016).

[12] Thong, P.H., Son, L.H.: A novel automatic picture fuzzy clustering method based on particle swarm optimization and picture composite cardinality, Knowl.-Based Syst., vol. 109, pp. 48–60 (2016).