Predict the Next Word in Sentence using Neural Networks

1]Rahul Marru
[1]Department of Data Science, rmarru@memphis.edu
2]Manivardhan Reddy Pindi
[2]Department of Data Science, mpindi@memphis.edu

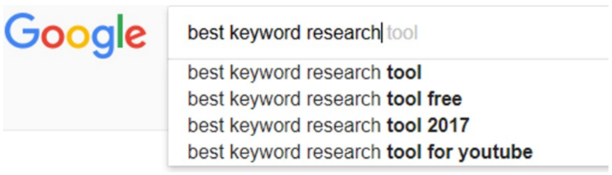
Abstract

The aim of the project is to predict the next word in a sentence; it is also known as "language modeling." It plays a major role in NLP applications like text summarization and the translation of questions and answers into various languages. In general, a language model will take as input a raw text corpus, and the probabilities of each word in the data will be calculated to find the similarity distance between them. This text corpus is split into two portions: 70% of training data and 30% of test data. For this problem, we are now solving it by using neural networks for language modeling. The three techniques, which are RNN (recurrent neural network), LSTM (long short-term memory), and Bi-LSTM (bi-directional long short-term memory), are combined with training data. It can be trained for up to 20 epochs and evaluated with test data to determine accuracies for models. Finally, using test cases as examples, it predicts the next word of the sentence. Note: The next word is either a word or the continuation letters of a word.

1 Introduction

As the task's title indicates, language modeling is a method of predicting the words that will appear in a sequence. It is a task that will predict which word comes next in the sequence. It is part of natural language processing and is used in deep learning. As a result, deep learning is an AI capability that allows it to make decisions based on a set of hidden patterns, similar to the human brain.

Text, audio, and video are some examples of the unstructured data used for unsupervised learning. The topic of "Neural Language Modeling" has so many real world implications. That is According to historical search data, Google search recommendations frequently use the forecast of the next word to automatically display searches that were comparable in the past. Not only in our everyday applications, but also in emails and chat messages, such as when we frequently enjoy messaging one another and discover that every time we try to enter a message, a suggestion pops up attempting to guess the next word we intend to type.



best keyword research tool
best keyword research tool
best keyword research tool free
best keyword research tool 2017
best keyword research tool for youtube

Figure 1: While Google Search it retrieves the next predicted word

Deep learning libraries such as TensorFlow and Keras, which require built-in models and layers, can be used.

2 Related Work

This problem was previously solved by authors Sourabh Ambulgekar, Sanket Malewadikar, Raju Garande, and Dr. Bharti Joshi, whose research found that NLP is a key component and has many applications. This work will be done using Tensorflow. This model was designed to predict at least 10 words in the shortest amount of time possible. There are excruciating disadvantages for the first model: due to their extensive short-term memory, RNNs are able to comprehend prior knowledge and foresee words. Then move forward with a new approach. LSTM In order to build a word, this method predicts a letter after another using letter-to-letter prediction. The accuracy for LSTM is 56%, which is better than RNN. It did a great job predicting the next word in the given sentence. Below are some of the test cases used by authors to solve this problem by using an LSTM neural network model rather than RNN model to predict the next word for the given input as a short sentence or text. (Ambulgekar et al., 2021).

Long sentences may be difficult to compose, but text prediction technology found in keyboards

1. it is hard enough hi i am sanket whats y
[’a ’, ’upiai ’, ’e ’, ’temtcid ’, ’ic ’]
2. it is which does not hurt us makes us str
[’ength ’, ’onger, ’, ’iver ’, ’ange ’, ’ungarity ’]
3. i am not sad that you lied to me, i am u
[’pon ’, ’nder ’, ’s ’, ’tility ’, ’ltimate ’]
4. those who were seen vibing were tho
[’ught ’, ’se ’, ’re ’, ’igh ’, ’m ’]
5. is though enough to remember my opinion
[’ and ’, ’, ’, ’s ’, ’. ’]
6. not a lack of effection , but a lack of
[’the ’, ’a ’, ’man ’, ’something ’, ’his ’]

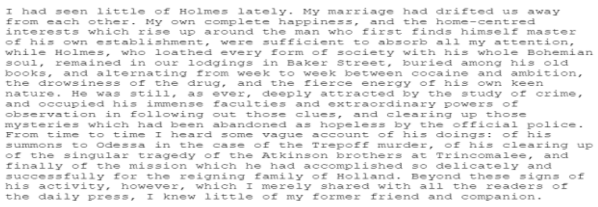
Figure 2: Test Cases - N(5) number of outputs

makes this easier. Next word prediction is also referred to as language modeling. Predicting the first word that will be stated is the current task. It is one of the primary functions of human language technology and has numerous applications. This method uses letter-to-letter prediction and claims to be able to anticipate a letter when it is used to form a word. The long short time memory equation may recognize previous material and foresee the words that might again be helpful to users when surrounding sentences. Word prediction algorithms will have to predict the word that will most likely be followed from a small set of initial text chunks. Researchers wish to simplify the process of immediate communications technology by suggesting appropriate terms to the consumer. The development and application of deep learning architectures led to the proposed convolutional neural networks (CNN), recurrent neural networks (RNN), and long short-term memory networks (LSTM). Other non-linear sequencing models, like LSTM, may be able to handle serialized data when analyzing prediction concerns. (Nayak and Kumar)

3 Data

The data corpus considered for this problem was collected from an English-language eBook about Project Gutenberg’s "The Adventures of Sherlock Holmes," written by author Arthur Conan Doyle, that contained various sub-contents like stories, documentation, and text data, which are necessary for our problem statement. It contains 581,884 words and 73 characters in the data corpus. In addition, we can use a variety of articles as text data for this problem.

Data corpus link: Source to access



I had seen little of Holmes lately. My marriage had drifted us away from each other. My own complete happiness, and the home-centred interests which rise up around the man who first finds himself master of his own establishment, were sufficient to absorb all my attention, while Holmes, who loathed every form of society with his whole Bohemian soul, remained in our lodgings in Baker Street, buried among his old books, and alternating from week to week between cocaine and ambition, the drowsiness of the drug, and the fierce energy of his own keen nature. He was still, as ever, deeply attracted by the study of crime, and occupied his immense faculties and extraordinary powers of observation in following out those clues, and clearing up those mysteries which had been abandoned as hopeless by the official police. From time to time I heard some vague account of his doings: of his summons to Odessa in the case of the Trepoff murder, of his clearing up of the singular tragedy of the Atkinson brothers at Trincomalee, and finally of the mission which he had accomplished so delicately and successfully for the reigning family of Holland. Beyond these signs of his activity, however, which I merely shared with all the readers of the daily press, I knew little of my former friend and companion.

Figure 3: Sample data from data corpus

4 Experiment Design

Following steps to implementing:-

- Load required packages and data from corpus.
- Feature Engineering.
- Building the model.
- Training, Evaluating and Testing the model.
- Predicting the next word for a sentence using best model.

Using Packages Keras, Tensorflow, Model – sequential Simple RNN, LSTM, Bidirectional, Dense. Activation layers, Text Feature engineering with text data is converting the string into numerical values. Applying train-test split and implementing the model with training data of 70%. Finding accuracy and loss with test data of 30%.

5 Methods

The three deep learning techniques used to solve this problem are RNN(Recurrent neural network) , LSTM(Long short-term memory) and Bi-LSTM (Bidirectional Long short-term memory).

Recurrent neural network Feed-forward neural networks with a focus on temporal modeling are known as RNNs. RNNs are characterized by their capacity to transmit data over time steps. An additional matrix for connections between time steps is a feature of RNNs’ structural design that encourages training in the temporal domain and makes use of the sequential character inputs. RNNs are trained to produce results in which the predictions at every time step are based on the current input and data from the prior time steps. The time series domain can be used to analyze input using RNNs. Data in one timestep are related to elements in earlier time steps, thus data in this domain are ordered and context-sensitive. (Singh et al., 2022).

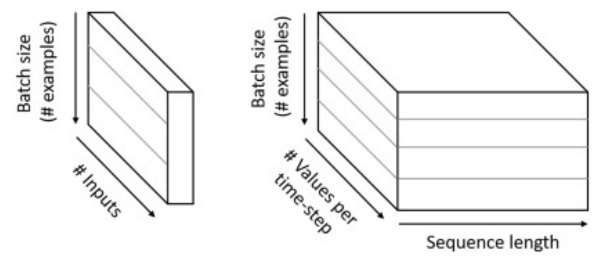


Figure 4: Diagram for showing the difference between normal and RNN neural network

RNNs bring the concept of recurring connections, which was previously stated. With this kind of wiring, the outcome of a hidden layer neuron is reconnected to that neuron's feed stream. The RNN workflow is depicted visually to explain recurrent connections below.

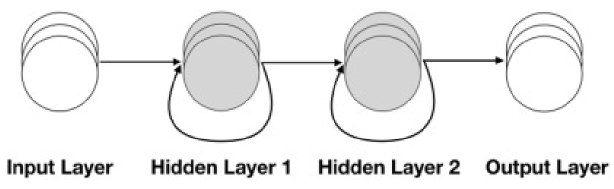


Figure 5: Demonstrate the Recurrent connections flow

The input from the hidden layer at one instant can be used as the input for the hidden units at a later point in time in RNNs because the hidden layers have a second self-looped connection. (Sharma et al., 2022)

$$o_t = f(\ h_t, \theta) \tag{1}$$

$$h_t = g(\ h_{t-1}, x_t, \theta) \tag{2}$$

Whereas,
θ – **means**
represents the networks bias and weights The term represents in equations are Hidden state, Input and Output.

Limits of RNN: The process of modeling statistical language using recurrent neural networks.Although it has been demonstrated that these methods performed well on this problem, frequently better than other looping approaches, they have a range of significant shortcomings, such as an extremely lengthy training period and a limitation on the number of language models that may be used in practice. In an effort to fix these issues, recurrent neural network models have recently been extended. (De Mulder et al., 2015)

Long short-term memory Algorithm The term "long short-term memory" (LSTM) refers to a particular type or improved form of the recurrent neural network (RNN) architecture used in the deep learning field. The LSTM is built to eliminate long-term dependence and contains feedback links. It can process entire knowledge sequences as well as single data items. For example, LSTM can be used for tasks like unstructured data, connected recognizing patterns, speech recognition, anomaly detection in network traffic, or IDSs (intrusion detection systems). The four major components of a typical LSTM unit are the cell, the input gate, the output gate, and the forget gate. The three gates normalize the flow of data or information into and out of the cell by allowing the cell to retain values over a range of arbitrary time intervals.

LSTM networks are a useful tool for classifying, analyzing, and making predictions based on time series data because critical events in a time series usually occur after delays of varying length.LSTMs were created to address the "vanishing gradient" issue that might arise when regular RNNs are being trained.Due to its relative insensitivity to gap length, LSTM frequently outperforms RNNs, hidden Markov models, and other sequence learning methods. The cell structure of LSTM is composed of four stages.The top horizontal line going across a cell denotes cell status. The LSTM may precisely control structures called "gates" that either exclude or include information in the cell state. The structure of the cell resembles a series. (Shende and Thorat, 2020)

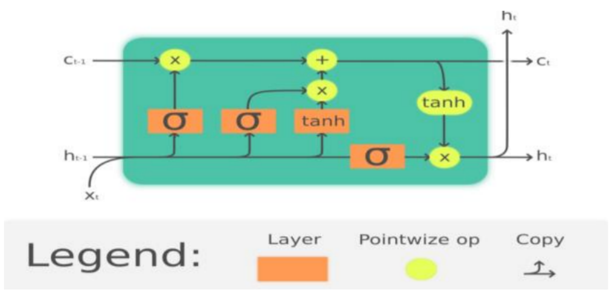


Figure 6: LSTM cell

Whereas, below equations represents the forget gate, input gate and output gate as well as cell state. Each gate's input weight, recurring weight, and bias are denoted by the letters W, h, and b, respectively. Additionally, sigmoid function and matrix multiplication are indicated by the   and operators (·) (Huang et al., 2021)

$$h_t = \sigma(\ w_f.[h_{t-1}, x_t] + b_f) \quad (3)$$

$$i_t = \sigma(\ W_i.[h_{t-1}, x_t] + b_i) \quad (4)$$

$$\tilde{c}_t = tanh(\ W_c.[h_{t-1}, x_t] + b_c) \quad (5)$$

$$c_t = f_t * c_t + i_t * \tilde{c}_t \quad (6)$$

$$o_t = \sigma(\ W_0.[h_{t-1}, x_t] + b_0) \quad (7)$$

$$h_t = o_t * tanh(\ c_t) \quad (8)$$

The **Limits of LSTM technology** enables the storage of data over arbitrary time delays and the long-term propagation of error signals.In theory, by giving them proper, inversely signed inputs, the network could use other memory cells to reset cells with pointless data. The difficulty in learning the exact values required to reset the state of the cell prevents this from working successfully in practice. The inverse of the output squashing function h would need to be calculated in order to achieve this, among other things. Typically, when a new training sequence begins, the network won't instantly return to a neutral state.(Gers et al., 2000)

Bidirectional LSTM When employing BiLSTM networks, the two LSTM neural network parameters are separate from one another and use the same sentence word embeddings.Using the input vector xt and the previous hidden state fht-1, the forward LSTM calculates the hidden state fht at each time step t, while the reverse LSTM generates the hidden state bht using the input vector xt and the preceding opposite hidden state bht-1. The final hidden state of the BiLSTM model is obtained by concatenating the vectors from two directions.

The bidirectional LSTM (BiLSTM) is made up of two separate LSTMs that collect word annotations and combine sentiment analysis from both sides of a sentence. (Yao, 2017)

In essence, the neurons of a typical RNN are divided into two different ways, and this is how BRNN works. One is for reverse states or the underside of time, while the other is for retrograde states or the positive side of time.The diagram below depicts the BiLSTM structure.Data from the past and future of the current time frame can be used as input by using two time directions. Standard RNN, however, demands delayed in order to include future data. (Sharfuddin et al., 2018)

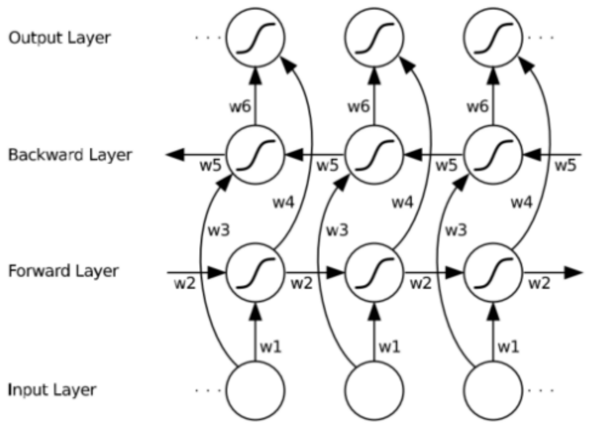


Figure 7: Bi-LSTM Neural Network Model

$$h_t = [fh_t, bh_t] \quad (9)$$

6 Experiments

NLP and neural networks were combined during the experiments. Pre-processing, feature engineering, and text classification are used to separate the data corpus into X inputs and Y outputs, as was previously described. Proceed with X and Y split into training and testing data.

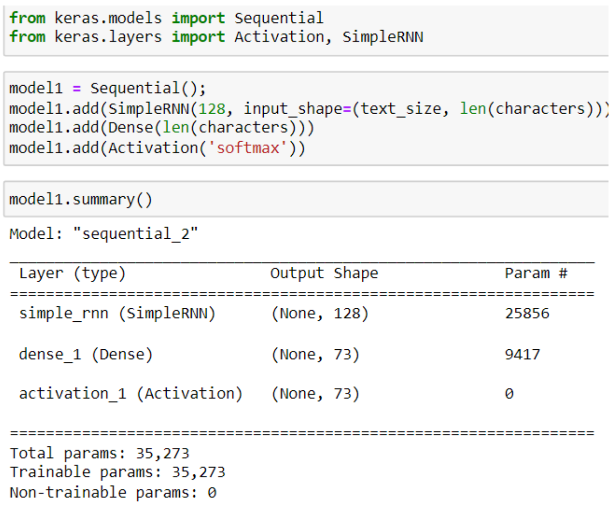
The models are then developed using the necessary libraries, hidden layers, activation layer, and optimizer, all of which are crucial for neural network models.

Firstly, the RNN model primarily used the built-in Sequential package in Keras to build the model. The hidden layers approached for this are simple RNN layer and Softmax activation function. The model is trained with training data of 70% up to 20 epcohs of observations, respectively, using the "Adam optimizer." In this process, we will get the outcomes like loss function and accuracy from each stage of epcohs observations.

Table 1: The loss and accuracy of the RNN model with each epoch

Epochs	**Loss function%**	**Accuracy%**
$5^{th} Epoch$	2.68	0.25
$10^{th} Epoch$	2.54	0.27
$15^{th} Epoch$	2.83	0.21
$20^{th} Epoch$	3.00	0.17

Secondly, the LSTM model as mentioned pre-



```
from keras.models import Sequential
from keras.layers import Activation, SimpleRNN
```

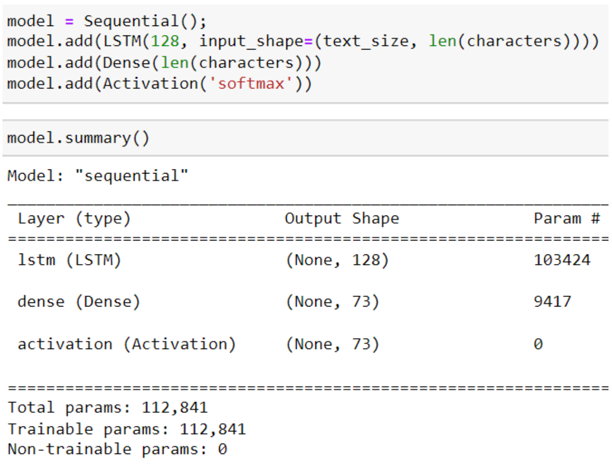
```
model1 = Sequential();
model1.add(SimpleRNN(128, input_shape=(text_size, len(characters)))
model1.add(Dense(len(characters)))
model1.add(Activation('softmax'))
```

```
model1.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 simple_rnn (SimpleRNN)      (None, 128)               25856

 dense_1 (Dense)             (None, 73)                9417

 activation_1 (Activation)   (None, 73)                0

=================================================================
Total params: 35,273
Trainable params: 35,273
Non-trainable params: 0
```

Figure 8: Recurrent Neural Network Model

viously it uses the built-in Keras package to build the model. The model as implemented with hidden layers has dense, LSTM, and softmax activation functions. Now, model training is done for up to 20 epochs, like an RNN with the "RMSprop" optimizer along with "categorical_crossentropy" for loss.



```
model = Sequential();
model.add(LSTM(128, input_shape=(text_size, len(characters))))
model.add(Dense(len(characters)))
model.add(Activation('softmax'))
```

```
model.summary()
```

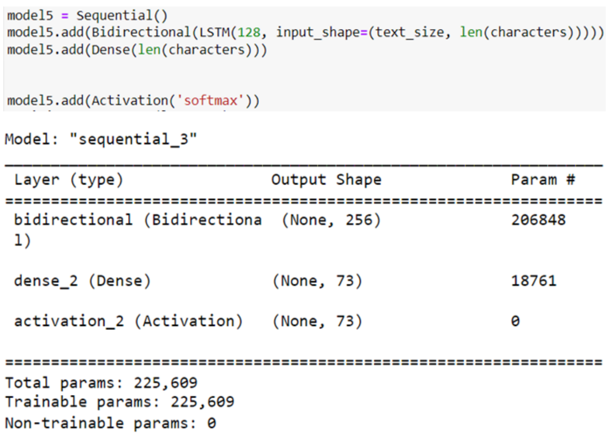
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 128)               103424

 dense (Dense)               (None, 73)                9417

 activation (Activation)     (None, 73)                0

=================================================================
Total params: 112,841
Trainable params: 112,841
Non-trainable params: 0
```

Figure 9: Long Short-Term Memory Model

Table 2: The loss and accuracy of the LSTM model with each epoch

Epochs	**Loss function%**	**Accuracy%**
5th $Epoch$	1.45	0.56
10th $Epoch$	1.33	0.59
15th $Epoch$	1.29	0.60
20th $Epoch$	1.26	0.61

Finally, the bidirectional model is built the same as above with Keras libraries, with hidden layers of dense, bidirectional, softmax activation functions and other layers. The optimizer is "RMSprop." The model is trained under 10 epcohs with categorical cross-entropy to compile the loss along with metrics accuracy to calculate the performance of the model.



```
model5 = Sequential()
model5.add(Bidirectional(LSTM(128, input_shape=(text_size, len(characters)))))
model5.add(Dense(len(characters)))


model5.add(Activation('softmax'))
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 bidirectional (Bidirectiona  (None, 256)              206848
 l)

 dense_2 (Dense)             (None, 73)                18761

 activation_2 (Activation)   (None, 73)                0

=================================================================
Total params: 225,609
Trainable params: 225,609
Non-trainable params: 0
```

Figure 10: Bi-directional Long Short-Term Memory Model

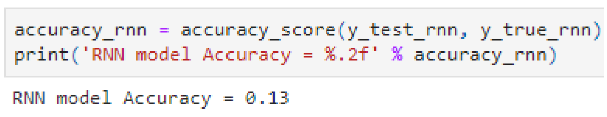
Table 3: The loss and accuracy of the Bi-LSTM model with each epoch

Epochs	**Loss function%**	**Accuracy%**
2st $Epoch$	0.84	1.00
4th $Epoch$	7.11	0.51
6th $Epoch$	3.09	0.51
8th $Epoch$	0.08	1.00
10th $Epoch$	0.04	1.00

7 Result

For all the observations of three neural network models. The LSTM model has better performance than the other RNN and Bi-LSTM models. Among these RNN, got low performance standards, but Bi-LSTM is nearly identical to the LSTM algorithm. The main criteria for considering LSTM are that it is consistent with its loss and accuracy through each epoch. Whereas in Bi-LSTM, it shows various variations in loss and accuracy scores during each epoch.

Accuracy score of RNN Model after training the model, the remaining 30% of test data is used for the RNN model evaluation. When using a model to predict the outcome of test data, compare the predicted and actual outcomes to determine the accuracy, which represents the model's performance.



```
accuracy_rnn = accuracy_score(y_test_rnn, y_true_rnn)
print('RNN model Accuracy = %.2f' % accuracy_rnn)

RNN model Accuracy = 0.13
```

Figure 11: Accuracy of RNN Model

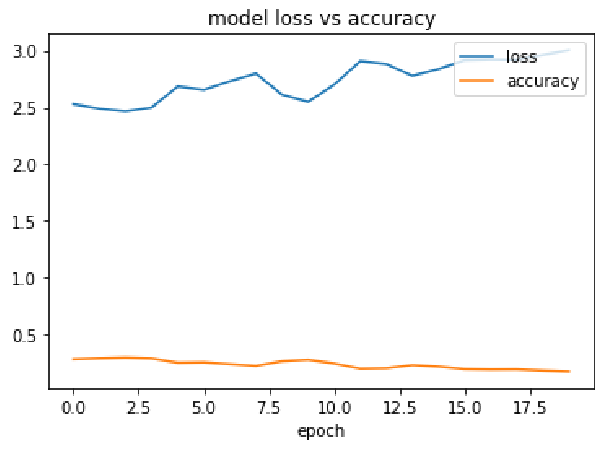
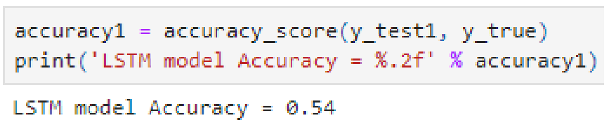


Figure 12: Comparison of Loss and Accuracy on graph plot for RNN Model

Accuracy score of LSTM Model the same approach that was mentioned earlier for the RNN model also applies to the LSTM model. The performance of a model is evaluated by remaining 30% of the test data to predict the outcome with an LSTM model. Then compare the actual outcome with the predicted outcome to calculate the accuracy.



```
accuracy1 = accuracy_score(y_test1, y_true)
print('LSTM model Accuracy = %.2f' % accuracy1)

LSTM model Accuracy = 0.54
```

Figure 13: Accuracy of LSTM Model

Accuracy score of Bi-LSTM Model the accuracy is calculated by compare of predicted outcome and actual outcome. The prediction outcome is generated by Bi-LSTM model on left over 30% of test data.

8 Test Cases

Predicting the next word for the given input sentence using the best model of the three techniques is LSTM, which has better accuracy and loss score.

9 Conclusion

To summarize, LSTM is the model with the highest accuracy and consistency. The main challenges are that running the LSTM model takes more time;

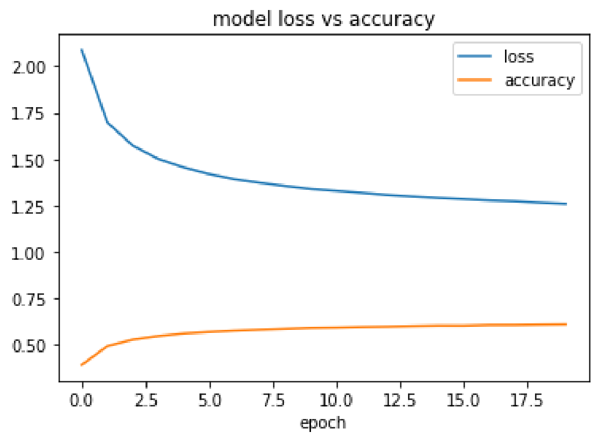
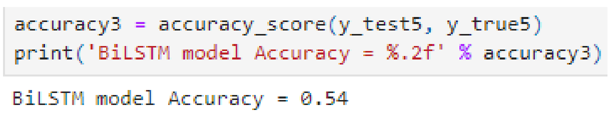


Figure 14: Comparison of Loss and Accuracy on graph plot for LSTM Model



```
accuracy3 = accuracy_score(y_test5, y_true5)
print('BiLSTM model Accuracy = %.2f' % accuracy3)

BiLSTM model Accuracy = 0.54
```

Figure 15: Accuracy of Bi-LSTM Model

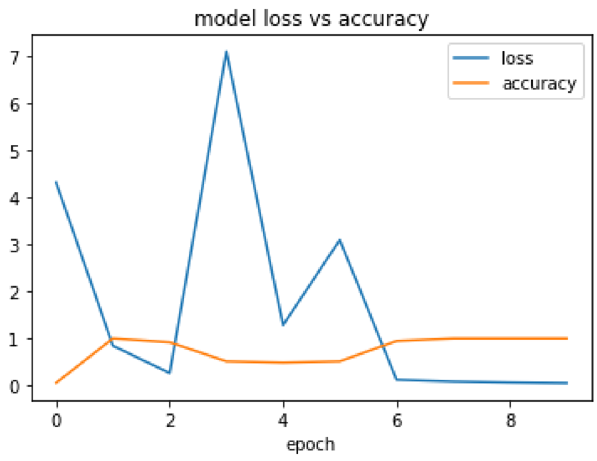
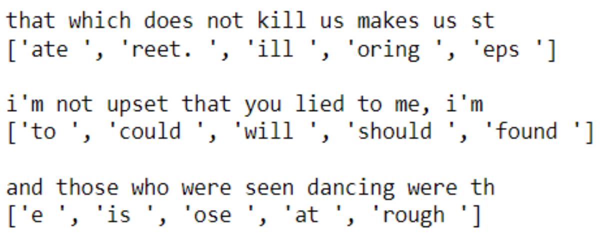


Figure 16: Comparison of Loss and Accuracy on graph plot for Bi-LSTM Model



```
that which does not kill us makes us st
['ate ', 'reet. ', 'ill ', 'oring ', 'eps ']

i'm not upset that you lied to me, i'm
['to ', 'could ', 'will ', 'should ', 'found ']

and those who were seen dancing were th
['e ', 'is ', 'ose ', 'at ', 'rough ']
```

Figure 17: 3 Test Cases with LSTM Model

however, if we train more, we can improve performance, and also if we increase the number of layers in these neural network models. We did a train-test split before applying models, and Bi-LSTM is new to our project, so it is almost as accurate or close to it when compared to previous baseline work. (Ambulgekar et al., 2021)

10 Contribution

Manivardhan Reddy Pindi :-

- Responsible for feature extraction (engineering) and Storing features and labels from the data corpus which is initially done.

- Fine tuning the model for better performance.

Rahul Marru :-

- Responsible for implementing the RNN, LSTM and Bi-LSTM model, which plays a major role in the project to find the accuracy and loss of each epoch of the model with 20 epochs. Along with citation suggestions for the baseline of the project.

- Verifying (Evaluating) and test the model by predicting the next few words for the given sentence. Try with model which is it get good performance than remaining models.

References

Sourabh Ambulgekar, Sanket Malewadikar, Raju Garande, and Bharti Joshi. 2021. Next words prediction using recurrent neuralnetworks. In *ITM Web of Conferences*, volume 40, page 03034. EDP Sciences.

Wim De Mulder, Steven Bethard, and Marie-Francine Moens. 2015. A survey on the application of recurrent neural networks to statistical language modeling. *Computer Speech & Language*, 30(1):61–98.

Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. 2000. Learning to forget: Continual prediction with lstm. *Neural computation*, 12(10):2451–2471.

Deqing Huang, Yuanzhe Fu, Na Qin, and Shibin Gao. 2021. Fault diagnosis of high-speed train bogie based on lstm neural network. *Sci. China Inf. Sci.*, 64(1):260–262.

Mr Chinmaya Nayak and Arvind Kumar. Next word prediction using machine learning techniques.

Abdullah Aziz Sharfuddin, Md Nafis Tihami, and Md Saiful Islam. 2018. A deep recurrent neural network with bilstm model for sentiment classification. In *2018 International conference on Bangla speech and language processing (ICBSLP)*, pages 1–4. IEEE.

Deepak Kumar Sharma, Mayukh Chatterjee, Gurmehak Kaur, and Suchitra Vavilala. 2022. Deep learning applications for disease diagnosis. In *Deep Learning for Medical Applications with Unique Data*, pages 31–51. Elsevier.

Supriya Shende and Samrat Thorat. 2020. Long short-term memory (lstm) deep learning method for intrusion detection in network security. *International Journal of Engineering Research and*, 9.

Eshan Singh, Nursulu Kuzhagaliyeva, and S Mani Sarathy. 2022. Using deep learning to diagnose preignition in turbocharged spark-ignited engines. In *Artificial Intelligence and Data Driven Optimization of Internal Combustion Engines*, pages 213–237. Elsevier.

Xianglu Yao. 2017. Attention-based bilstm neural networks for sentiment classification of short texts. In *Proc. Int. Conf. Inf. Sci. Cloud Comput*, pages 110–117.