International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 07 Issue: 04 | April - 2023 Impact Factor: 8.176 ISSN: 2582-3930

SUCCESSIVE WORD PREDICTION

Mr. Ch. Praneeth, Assistant Professor
Department of Information Technology
Prasad V Potluri Siddhartha Institute of Technology
Vijayawada, India
Email: chpraneeth@pvpsiddhartha.ac.in

Keerthi Veginati
Department of Information Technology
Prasad V Potluri Siddhartha Institute of Technology
Vijayawada, India
Email: keerthisriveginati72@gmail.com

Leela Sri Yarlagadda
Department of Information Technology
Prasad V Potluri Siddhartha Institute of Technology
Vijayawada, India
Email: yarlagadda.leelasri24@gmail.com

Sirisha Pidamarthi
Department of Information Technology
Prasad V Potluri Siddhartha Institute of Technology
Vijayawada, India
Email: sirishachinni363@gmail.com

Supritha Murugu
Department of Information technology
Prasad V Potluri Siddhartha Institute of Technology
Vijayawada, India
Email: murugusupritha@gmail.com

***Abstract*— Worldwide, people are using their mobile device, laptop, tablet, desktop, etc. for activities including messaging, emailing, banking, and social media connection anything else, including the media. The amount of time spent typing on these gadgets must be reduced. When the gadget can offer the user more possibilities for the potential next word for the currently written word, it can be accomplished. Also, it speeds up typing. In this paper, we propose and offer a comparative analysis of several models, including recurrent neural networks, stacked recurrent neural networks, long short-term memory networks (LSTM), and bi-directional LSTM, that provide a solution to the issue. Our main objective is to recommend which of the four models will forecast the following word for the supplied current word in the English language.**

Keywords: **Recurrent Neural Network (RNN), Long Short Term Memory (LSTM), Bi-Directional LST.**

I. INTRODUCTION

Artificial intelligence is the process of programming a machine to think and behave like a person by absorbing information from its surroundings and acting on it in accordance with that information. The purpose of neural networks is to mimic the human brain. It is made up of linkages that connect different neurons to build a network. Layers of linked units known as artificial neurons are used to form Artificial Neural Networks (ANNs), a combination of Artificial Intelligence and Neural Networks [1] [4]. There are input, output, and hidden layers in this network. There is only one input layer and one output layer. There could be one to many hidden layers. The complexity of the network will rise along with the number of hidden levels. This vast network might facilitate deep learning. When neurons are connected repeatedly in artificial neural networks, the network is known as a recurrent neural network (RNN) [1]. The initial output of an RNN is based on the input, and subsequent outputs are based on the preceding output. This creates a pattern in the data. RNNs have internal state and memory that aid in processing a variety of input sequences. Due to this, RNN is widely utilized in a variety of fields, including speech recognition, protein structure prediction, handwriting recognition, etc. Recurrent connections enable neural networks to function better by allowing for the knowledge of the dependencies between sequences of data. The issue arises when RNN is training the data [2]. The RNN leads encounter the vanishing-gradient problem during the training phase [4]. This is unable to store the data's long-term dependencies. A model called Long-Short Term Memory (LSTM) RNNs is created to precisely address this issue. It significantly lessens the vanishing-gradient problem. One "forget gate" and input and output gates make up this network layout. Input and output gates control the data flow to the buried neural layer and protect the features that were retrieved from earlier time steps. It should be noted that the internal numerical value of the LSTM model may increase dramatically for continuous data sequences [2]. The memory and the forget gate offer a solution to the vanishing-gradient issue. Backpropagation to the same network is a method for

correcting the discovered fault. Backpropagation enables LSTM to learn tasks requiring memories of discrete time events that occurred thousands of time steps or more in the past. Even events with lengthy delays can be handled with LSTM. Bidirectional LSTM (BLSTM), which interprets the input in the opposite manner, is created by stacking two LSTM models on top of one another. The input will be interpreted from right to left by one LSTM and from left to right by the other. The hidden state of both LSTMs determines the output. BLSTM network model is hence more effective than unidirectional LSTM model. To enhance the functionality of unidirectional RNN, two RNN can be placed on top of one another to create stacked RNN [3] [4].

II. TECHNIQUES

Python Libraries

The libraries of python used here are numpy, pandas, matplotlib. The numpy is to operate the arrays and matrices, pandas are used to deal with the ML tasks and data analytics and also to load the data from the csv file to the data frame of panda. Matplotlib is used to have static, animated and interactive visualizations for our data.

LSTM

LSTM stands for long short-term memory networks, used in the field of Deep Learning. It is a variety of recurrent neural networks (RNNs) that are capable of learning long-term dependencies, especially in sequence prediction problems. LSTM is a special kind of RNN, which shows outstanding performance on a large variety of problems.

BILSTM

Bidirectional LSTM (BiLSTM) means both ways of input are accepted, and both sides of the information can be used. With the help of BiLSTMs, the network has access to more information, which benefits the algorithm's context (e.g., knowing what words immediately follow and precede a word in a sentence).

RNN

An artificial neural network that employs sequential data or time series data is known as a recurrent neural network (RNN). Recurrent neural networks identify patterns in data and utilize them to anticipate the following most likely scenario. Deep learning and the creation of models both utilize RNNs.

Stacked RNN

Feeding the output of one RNN layer to another RNN layer is the definition of stacking RNNs. The RNN layers have the ability to produce sequences (i.e., output at every time step), which can then be fed into the following RNN layer just like any other input sequence.

Stacked RNNs are intended to replace regular RNNs and build complicated models. The output of the previous layer is fed to the subsequent layer, with each layer being fully recurrent. Another fully recurring RNN is found at the next level.

III. METHODOLOGY

The article file's headlines are used. A preprocessing input is provided for each headline. Tokenization is then completed. Each word is given a random number, the longest headline among them is chosen as the vector length, and all the headlines are then ordered in a vector sequence using the N-gram sequence with left padding. The input is given to different models, and the output is used as training data. The test data is provided with a range of lengths, and the anticipated output length is also provided. Several models' loss and accuracy are examined.

RNN is extended by the Long Short-Term Memory (LSTM). The hidden layer, a recurring layer that comprises memory cells, has memory blocks in it. To maintain the network's temporal state, it has recurrent connections. Also, the network features unique gates, multiplicative units that govern the information flow throughout the network.

The input and output gates, respectively, are in charge of controlling the input and output activations. The LSTM network has an additional forget gate that it can use to enable or disable its own state.

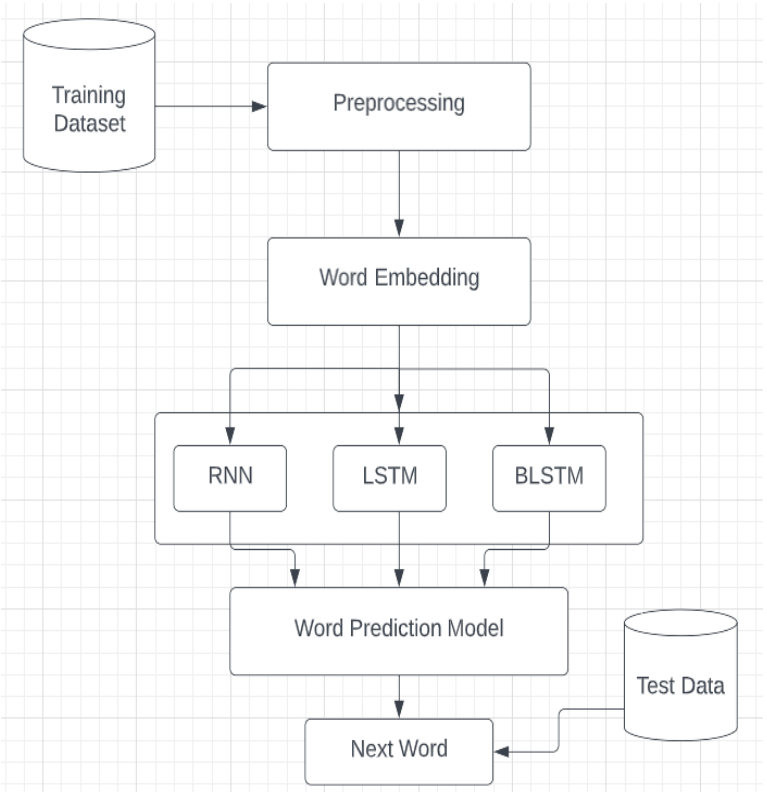

Fig.1 Working of Proposed System Architechture

The timing of the outputs is learned by LSTM using additional connections.

By dividing the neurons of a typical LSTM into two positive time directions and two negative time directions, the

Bidirectional LSTM (Bi-LSTM) lengthens the LSTM. The forward state is indicated by the positive time direction, and the reverse direction indicates the backward state. The preceding and subsequent states are inputs for this two-time direction. The LSTM cell in our model is bidirectional to maintain information flow in both directions while still having the same Deep LSTM structure.

IV. RESULT

```
text_collection = ['deep convolutional', 'simple and effective', 'a nonconvex', 'the

next_word(text_collection)
```

```
1/1 [==============================] - 0s 20ms/step
deep convolutional --> deep convolutional performance
NEXT WORD: performance

1/1 [==============================] - 0s 19ms/step
simple and effective --> simple and effective acceleration
NEXT WORD: acceleration

1/1 [==============================] - 0s 32ms/step
a nonconvex --> a nonconvex dataset
NEXT WORD: dataset

1/1 [==============================] - 0s 23ms/step
high computational --> high computational networks
NEXT WORD: networks

1/1 [==============================] - 0s 19ms/step
feedforward --> feedforward data
NEXT WORD: data
```

Fig.2 Prediction of Next Word from the given Dataset using BILSTM.

```
[13] # Training the model.

model.fit(X_train, y_train, batch_size=512, shuffle=True, epochs=20, validation_data=(X_test, y_test), callbacks=[LambdaCallb
```

```
Epoch 1/20
11/11 [==============================] - 3s 234ms/step - loss: 7.6565 - acc: 0.1046 - val_loss: 7.0355 - val_acc: 0.1189
Epoch 2/20
11/11 [==============================] - 2s 160ms/step - loss: 5.9598 - acc: 0.0626 - val_loss: 4.5251 - val_acc: 0.0444
Epoch 3/20
11/11 [==============================] - 2s 139ms/step - loss: 4.2150 - acc: 0.0506 - val_loss: 4.0581 - val_acc: 0.1067
Epoch 4/20
11/11 [==============================] - 1s 139ms/step - loss: 3.9599 - acc: 0.1283 - val_loss: 3.8324 - val_acc: 0.2456
Epoch 5/20
11/11 [==============================] - 1s 137ms/step - loss: 3.7218 - acc: 0.2274 - val_loss: 3.5675 - val_acc: 0.2928
Epoch 6/20
11/11 [==============================] - 1s 136ms/step - loss: 3.4172 - acc: 0.3326 - val_loss: 3.2226 - val_acc: 0.4300
Epoch 7/20
11/11 [==============================] - 1s 126ms/step - loss: 3.0428 - acc: 0.5835 - val_loss: 2.8177 - val_acc: 0.6767
Epoch 8/20
11/11 [==============================] - 1s 124ms/step - loss: 2.6140 - acc: 0.7156 - val_loss: 2.3662 - val_acc: 0.8339
Epoch 9/20
11/11 [==============================] - 2s 205ms/step - loss: 2.1514 - acc: 0.9146 - val_loss: 1.9071 - val_acc: 0.9522
Epoch 10/20
11/11 [==============================] - 2s 168ms/step - loss: 1.6925 - acc: 0.9648 - val_loss: 1.4705 - val_acc: 0.9889
```

Fig.3 Accuracy and Loss of the model using LSTM.

```
[17]: model.fit(X, y, epochs=150, batch_size=64, callbacks=[checkpoint, reduce, tensorboard_Visualization])
```

```
Train on 3889 samples
Epoch 1/150
3712/3889 [===========================>..] - ETA: 0s - loss: 7.8752
Epoch 00001: loss improved from inf to 7.87560, saving model to nextword1.h5
3889/3889 [==============================] - 5s 1ms/sample - loss: 7.8756
Epoch 2/150
3648/3889 [===========================>..] - ETA: 0s - loss: 7.8587
Epoch 00002: loss improved from 7.87560 to 7.86009, saving model to nextword1.h5
3889/3889 [==============================] - 1s 331us/sample - loss: 7.8601
Epoch 3/150
3648/3889 [===========================>..] - ETA: 0s - loss: 7.8187
Epoch 00003: loss improved from 7.86009 to 7.81623, saving model to nextword1.h5
3889/3889 [==============================] - 1s 372us/sample - loss: 7.8162
Epoch 4/150
3648/3889 [===========================>..] - ETA: 0s - loss: 7.6399
Epoch 00004: loss improved from 7.81623 to 7.63961, saving model to nextword1.h5
3889/3889 [==============================] - 1s 327us/sample - loss: 7.6396
Epoch 5/150
3648/3889 [===========================>..] - ETA: 0s - loss: 7.4280
Epoch 00005: loss improved from 7.63961 to 7.42898, saving model to nextword1.h5
3889/3889 [==============================] - 1s 363us/sample - loss: 7.4290
Epoch 6/150
3648/3889 [===========================>..] - ETA: 0s - loss: 7.2234
Epoch 00006: loss improved from 7.42898 to 7.23395, saving model to nextword1.h5
3889/3889 [==============================] - 1s 335us/sample - loss: 7.2339
```

Fig.4 Accuracy and Loss of the model using RNN.

```
89/89 [==============================] - 7s 24ms/step - loss: 7.2391 - accuracy: 0.0538
Epoch 2/250
89/89 [==============================] - 2s 24ms/step - loss: 6.7368 - accuracy: 0.0546
Epoch 3/250
89/89 [==============================] - 2s 25ms/step - loss: 6.7163 - accuracy: 0.0546
Epoch 4/250
89/89 [==============================] - 2s 24ms/step - loss: 6.6181 - accuracy: 0.0546
Epoch 5/250
89/89 [==============================] - 2s 24ms/step - loss: 6.5013 - accuracy: 0.0554
Epoch 6/250
89/89 [==============================] - 2s 24ms/step - loss: 6.3722 - accuracy: 0.0619
Epoch 7/250
89/89 [==============================] - 2s 24ms/step - loss: 6.2853 - accuracy: 0.0691
Epoch 8/250
89/89 [==============================] - 2s 25ms/step - loss: 6.2179 - accuracy: 0.0714
Epoch 9/250
89/89 [==============================] - 2s 26ms/step - loss: 6.1610 - accuracy: 0.0744
Epoch 10/250
89/89 [==============================] - 2s 24ms/step - loss: 6.1139 - accuracy: 0.0760
Epoch 11/250
89/89 [==============================] - 2s 24ms/step - loss: 6.0669 - accuracy: 0.0774
```

Fig.5 Accuracy and Loss of the model using CNN.

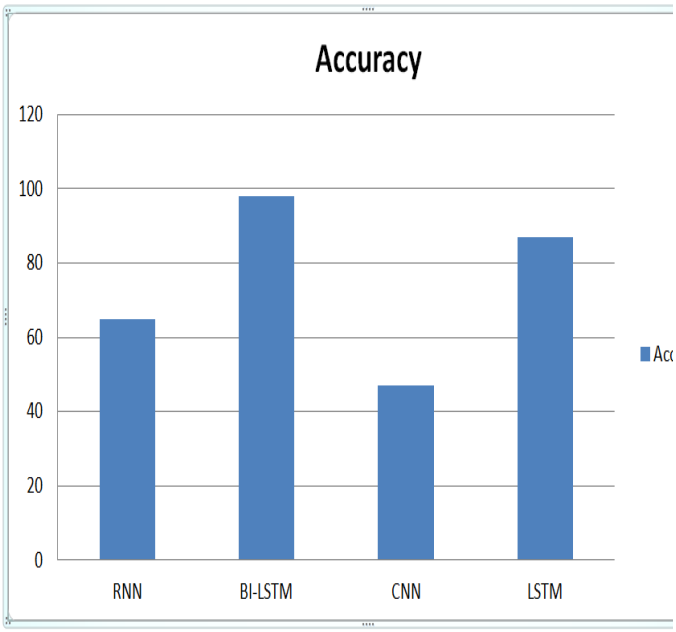

Fig.6 Comparison of Different Models.

V. Scope For Future Use

Future models that could be more accurate than BILSTM include those from RNN and LSTM. Short-term memory issues, such as gradients that vanish, explode, or become smaller as a model back propagates through time, are resolved by LSTMs. The LSTM has gates that can control what is remembered or discarded. As a result, it gains the ability to use pertinent information to generate prediction.

VI. CONCLUSION

In this study, we compared a range of sequential models with a range of activation and loss functions. The Bidirectional LSTM exhibits the lowest loss in this study when compared to other models for the training dataset. According to our study, RNN predicts the following word with an accuracy of 60% and a loss of 40%, Stacked Recurrent Neural Network **(**RNN**)** with an accuracy of 62% and a loss of 38%, LSTM with an accuracy of 64% and a loss of 36%, and Bidirectional LSTM with an accuracy of 72% and a loss of 28%. Cross Entropy and Bidirectional LSTM Combined Bidirectional LSTM with Cross Entropy performs worse than divergence. The examination of different models and optimizers will be used in future projects.

VII. REFERENCES

1) Bharti Joshi, Sourabh Ambulgekar, Raju Garande and Sanket Malewadikar, “Next word prediction using Recurrent Neural Networks”, ResearchGate January 2020.

2) Joel Stremmel and Arjun Singh, “Pretraining Federated Text Models for Next Word Prediction”, American Journal of Engineering Research (AJER), August 2020.

3) Jona JB, “Next Word Prediction using CNN”, International Journal of Creative research Thoughts (IJCRT), Volume 9 December 2021.

4) Sanjeev Thakur and Milind Soam, “Next Word Prediction using Deep Learning”, Institute of Electrical and Electronics Engineering (IEEE) March 2022, 12th International Conference on Data Science and Engineering, January 2022.

5) Ch. Praneeth, N. V. SanjanaNaidu, M. KondaReddy, P. Sirisha, Sk. Aashik, “The Tradeoff Between ML and DL Techniques”, International Journal for Innovative Engineering and Management Research (IJIEMR), Volume 12 February 2023.