2021 International Conference on Computer Communication and Informatics (ICCCI -2021), Jan. 27-29, 2021, Coimbatore, INDIA

Word Level LSTM and Recurrent Neural Network for Automatic Text Generation

Harsha Vardhana Krishna Sai Buddana
B.Tech Scholar,
Department of Electronics and Communication Engineering,
Marri Laxman Reddy Institute of Technology and Management,
Hyderabad,Telangana, India
harshabuddana@gmail.com

Surampudi Sai Kaushik
B.Tech Scholar,
Department of Electronics and Communication Engineering,
Marri Laxman Reddy Institute of Technology and Management,
Hyderabad,Telangana, India
surampudisaikaushik@gmail.com

PVS.Manogna
B.Tech Scholar,
Department of Electronics and Communication Engineering,
Marri Laxman Reddy Institute of Technology and Management,
Hyderabad,Telangana, India
pvs.manogna3020@gmail.com

Shijin Kumar P.S
Associate Professor,
Department of Electronics and Communication Engineering,
Marri Laxman Reddy Institute of Technology and Management,
Hyderabad,Telangana, India
shijinkumarps@yahoo.com

2021 International Conference on Computer Communication and Informatics (ICCCI) | 978-1-7281-5875-4/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICCCI50826.2021.9402488

Abstract— **Sequence prediction problems have been a major problem for a long time. Recurrent Neural Network (RNN) has been a good solution for sequential prediction problems. This work aims to create a generative model for text. Even though, RNN has its own limitations such as vanishing and exploding gradient descent problems, and inefficiency to keep track of long-term dependencies. To overcome these drawbacks, Long Short Term Memory (LSTM) has been a path-breaking solution to deal with sequential data and text data in particular. This paper delineates the design and working of text generation using word-level LSTM-RNN.**

Keywords— Long-Short term memory, recurrent neural network, text generation, deep learning, natural language generation.

I. Introduction

The proliferation of computational intelligence in the late 19th century, led to a surge in the development of complex and more efficient computer algorithms that abated some of the tedious tasks in many real world problems [1]. Today these technologies are omnipresent and are applied in many sectors of the industry. Even though, they are proven effective in most cases, there exist problems in the real world applications. This paved a path for the evolution of deep learning, which enabled computers to learn, understand and analyze themselves about its surrounding world.

There is no need for any human interaction in deep learning as the computer once trained, learns from its experience and produces intended results in a quicker and efficient manner. Many deep learning techniques are being used in some of the major industrial sectors such as robotics, defense, agriculture, etc.

Sequential data [2] in particular is very volatile in nature and is often expected to have many dependencies. To handle such data, special types of neural networks have been deployed and are proven effective in dealing with such exceptions. Neural networks work on the basic principle of the human brain neurons. These consist of nodes that operate over various activation functions that take in some inputs that result in a weighted sum of these inputs. One such type of neural network known as RNN is widely used solving data prediction problems. The major application of this approach is to generate text and stories for web contents.

II. Methodology

A. Recurrent Neural Network (RNN)

RNN [3] employs feedback connections that make them more powerful than feed forward neural networks. This establishment of feedback provides memory that administers an advantage in handling sequential data. Deep Neural Networks (DNN) [4], on the other hand, operate on fixed-sized windows of frames and provide restricted temporal modeling that may not be suitable for long-term dependencies. Fig.1 depicts the structure of RNN architecture.

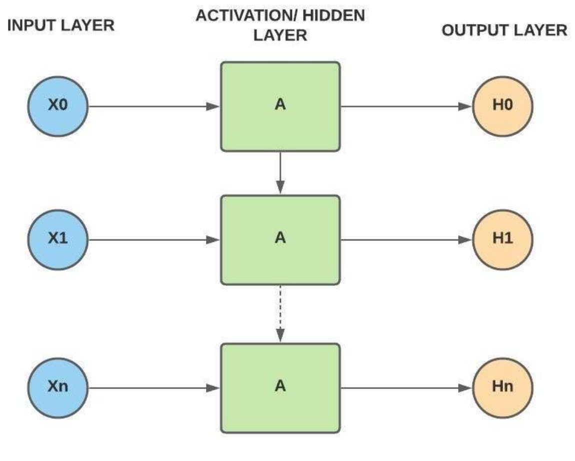


Fig.1. RNN Architecture

Due to this limitation, RNN's dynamic exploitation of the constantly changing data by considering the past input sequence rather than a static present one offers more flexibility and efficacy. In particular, LSTM-RNN [5] is an excellent tool for employing in sequential data applications especially while dealing with text data..

B. *Long Short Term Memory (LSTM)*

In order to abate the gradient diminishing problem that occurs in RNN, LSTM [6] is introduced. The LSTM retains long-term information through the back propagation of highly relevant information. Three gates namely, input, forget and output gates are present in the LSTM. These are connected to the cell state, hidden state and the input. The gates are driven by a dense sigmoid activation function. A simple RNN cell and the output cell state are driven by dense squashing (*tanh*) activation function. At each time stamp, the gates aid in determining the relevance of past information which are relevant to retain and those need to be forgotten.. Addition of these gates in LSTM aids in preserving history of longer sequences. This is an added advantage in many NLP applications. The architecture of proposed LSTM-RNN is depicted in Fig. 2.

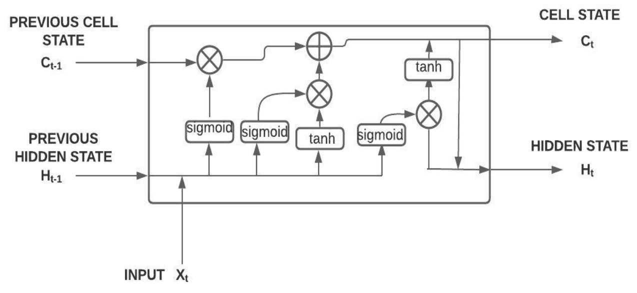


Fig.2. Architecture of LSTM

Mathematically, LSTM network is represented as,

At forget gate:

$$f_t = \sigma\left[K_f\left(h_{t-1}, x_t\right) + a_f\right] \tag{1}$$

$$C_t^f = C_{t-1} * f_t \tag{2}$$

At input gate:

$$i_t = \sigma\left[K_i\left(h_{t-1}, x_t\right) + a_i\right] \tag{3}$$

$$C_t' = tanh\left[K_c\left(h_{t-1}, x_t\right) + a_c\right] \tag{4}$$

$$C_t^i = C_t' * i_t \tag{5}$$

Cell State:

$$C_t = C_t^f + C_t^i \tag{6}$$

At output gate:

$$O_t = \sigma\left[K_o\left(h_{t-1}, x_t\right) + a_o\right] \tag{7}$$

$$h_t = O_t * tanh\left(C_t\right) \tag{8}$$

where, f_t, i_t, and O_t are the forget, input and output matrices respectively. K_f, K_i, K_c and K_o are the weight matrices for the respective gates. C_t is the total cell state for the network. It is a summation of cell states at the forget gate and the input gate. h_t is the hidden state of the network.

C. *Text Generation using word-level LSTM*

The model is first fed with a sequence of seed words and the succeeding words will be predicted by the trained model [7]. This output is combined with the previously given word sequence and this new sequence is taken as a new input for further predictions. This process is done iteratively the process of shifting window to the next word for generating a complete text. In this work, we used word level training of the language model [8].

We selected a story as the dataset and it eases the process by generating text related to some specific domain knowledge. Greek Philosopher Plato's famous work, The Republic of Plato, is used in our work. All the data is cleaned and all the punctuations are removed in the pre-processing step. Sentences of fixed word length are used for training purposes so as to enhance computational memory space. Once the sequences are loaded, the data is split into separate training sequences based on new line occurrence. Word embedding layer takes in input a sequence consisting of integer values. Each word in the vocabulary is then mapped to a unique integer and the input sequences are encoded. This helps in predicting the next word by associating words that are present in the same mapping. We employed the Tokenizer Class in Keras API in order to implement the encoding. The tokenizer is first trained on the entire data set and finds all the unique words. It assigns an unique integer to all the trained words. After training, fit tokenizer encodes the training sequences and convert it into a list of integers. Each word is vectorized and indexed. The methodology is briefly explained in Fig. 3.

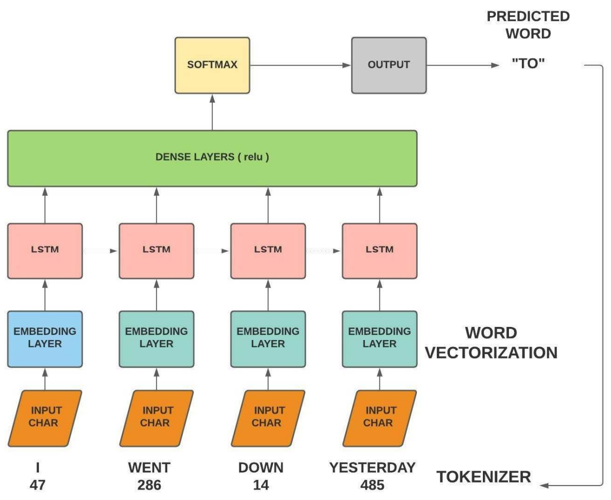


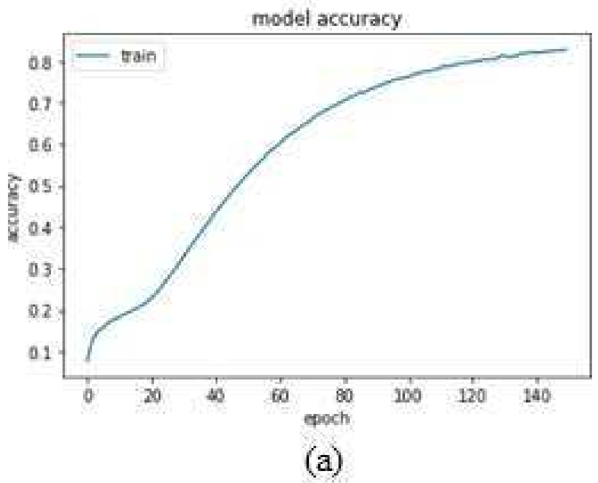
Fig.3. Text Generation using LSTM-RNN

After the model is trained, an input sequence of seed words is given as an input. These seed words help the network to initiate the process. A random line is considered from the input text and the model returns the words with highest probability and this predicted word is then appended back to the input seed sequence and is passed on to the next layer. This happens iteratively until the desired length is reached.

III. Results and Discussions

In general, LSTM models [9] are little different in validation perspective unlike the conventional RNN or CNN models. It is difficult to analyze the performance of the model based on test-train accuracy. It is a better option to test the model with real time data and analyze its performance. We trained and tested the model and were able to achieve

96% training accuracy and 80% validation accuracy. We employed 256 hidden layers for all the layers (LSTM as well as dense activation functions). Two LSTM layers are considered and two dense layers are used, one for *relu* and one for *softmax*. Before analyzing the results, it's imperative to know how the language model adapted itself over time [10]. The training accuracy and training loss curves over time demonstrates how the model adapted while undergoing the training process.



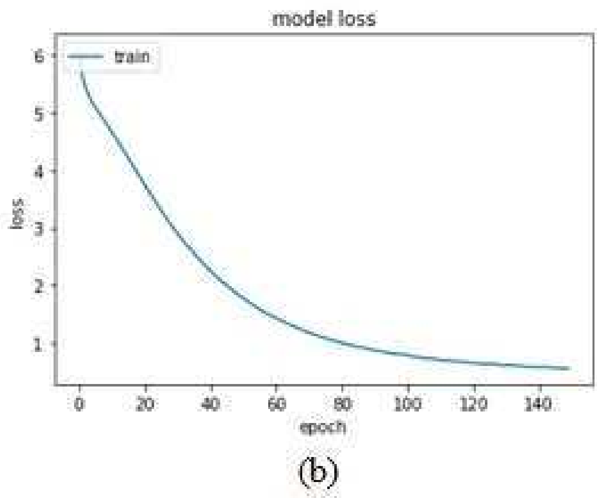


Fig.4. Model Performance (a) Training accuracy, (b) Training loss

Fig.4 (a) depicts the plot of accuracy Vs epochs (time) on training data. As depicted in the plot, we can see the training accuracy improved over time and reached over 70%. Although the model did not reach desired accuracy levels, it performs considerably well in generating the text. Fig.4 (b) depicts the plot of model loss on training dataset. We can see that the training loss decreases pretty well. The trained model was tested and was able to achieve 96% training accuracy and 80% validation accuracy with overall loss of 0.16. It is represented in Fig.5. The model is trained by tuning various training hyper parameters. Table I demonstrates accuracy of the model for various cases.

TABLE I. MODEL PERFORMANCE FOR VARIOUS CASES

No: of Layers	**Training Parameters**		**Accuracy(%)**	
	Batch Size	Epoch	Training	Validation
256	128	50	42	36
	128	75	57	46
	128	100	83	70
	128	150	96	80

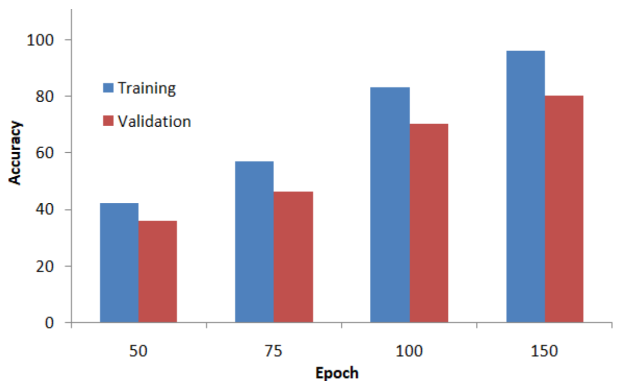


Fig.5. Evaluation of Model Performance

The trained model is tested on the Republic of Plato and the results are shown in Table II.

TABLE II. TEST RESULT OF PROPOSED TEXT GENERATOR

Input Seed Sequence	**Generated Text**
"entire life has a good report and carries off the prize which men have to bestow true and now you must allow me to repeat of the just the blessings which you were attributing to the fortunate unjust I shall say of them what you were saying of the others that"	i now say that you have seen injustice like men and again and again whether injustice is as much as in the case of the state still the good or the state and the spirit of the unjust intelligence when they were able to praise the way in which the nature of our protectors is made out to be a tyrant how they are.

The performance of proposed text generation system is compared with existing methods in terms of accuracy. The detailed comparison is explained in Table 3 and Fig. 6.

TABLE III. PERFORMANCE COMPARISON OF TEXT GENERATORS

Methodology	**Accuracy (%)**
Random	41
Baseline-Seq2Seq	62
WD-LSTM	76
Proposed (LSTM-RNN)	80

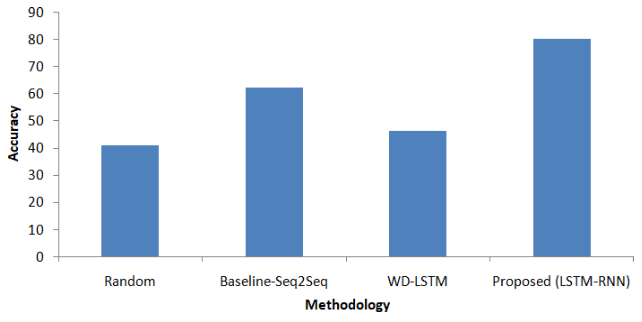


Fig. 6. Comparison of Performance

IV. Conclusion

This paper delineated a brief design and working of an LSTM-RNN Text generation model. LSTMs are the most widely used networks for language modelling. The main objective of this work is to train an LSTM model to generate text on a given dataset and analyze its performance. An accuracy of 80% has been achieved. This accuracy can further be increased by further reducing the training loss by tweaking the hyper parameters of the model. Experimental results shows that the proposed method outperform exixsting taraditional text generation systems.

References

[1] Z. Ghahramani, "Probabilistic machine learning and artificial intelligence," Nature, vol. 521(1), pp. 452-459, 2015.

[2] T. G. Dietterich , "Machine learning for sequential data: A review," In Joint IAPR international workshops on statistical techniques in pattern recognition, 2002, pp. 15-30, Springer, Heidelberg.

[3] I. Li, Q. Pan, S. Wang, T. Yang, E. Cambraia, "A generative model for category text generation," Information Sciences, vol. 450, pp. 301-315, 2018.

[4] W. Liu , "A survey of deep neural network architectures and their applications," Neurocomputing, vol. 234, pp. 11-26, 2017.

[5] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural computation, vol. 98, pp. 1735-1780, 1997.

[6] T. Zia and U. Zahid, "Long short-term memory recurrent neural network architectures for Urdu acoustic modeling," International Journal of Speech Technology, vol. 22, pp. 21-30, 2019.

[7] H. Deng, L. Zhang and L. Wang, "Global context-dependent recurrent neural network language model with sparse feature learning," Neural Computing and Applications, vol. 31, pp. 999-1011, 2019.

[8] D. Pawade, A. Sakhapara, M. Jain and N. Jain, "Story scrambler-automatic text generation using word level RNN-LSTM," International Journal of Information Technology and Computer Science, vol. 10, pp. 44-53, 2018.

[9] M. S. Islam, S. S. Abujar and S. A. Hossain "Sequence-to-sequence bangla sentence generation with LSTM recurrent neural networks," Procedia Computer Science, vol. 10, pp. 44-53, 2018.

[10] X. Dai, H. Yin and N. K. Jha, "Grow and Prune Compact, Fast, and Accurate LSTMs," IEEE Transactions on Computers, vol. 69, pp. 441-452, 2019.