Sentence Generation using LSTM Based Deep Learning

Sunanda Das, Sajal Basak Partha, Kazi Nasim Imtiaz Hasan
Department of Computer Science and Engineering
Khulna University of Engineering & Technology
Khulna-9203, Bangladesh
sunanda@cse.kuet.ac.bd, partha1607101@stud.kuet.ac.bd, nasim.cse.kuet@gmail.com

***Abstract*— Sentence generation serves the process of predicting relevant words in a specific sequence. The purpose of this research is to come up with a method for generating sentences while maintaining proper grammatical structure. Here, we have implemented a sentence generation system based on Long Short-Term Memory (LSTM) architecture. Our system generally follows the basics of word embedding where words from the dataset get tokenized and turned into vector forms. These vectors are then processed and passed through a Long Short-Term Memory layer. Successive words get generated from the system after each iteration. This process winds up generating relevant words to form a sentence or a passage. The results of the system are pretty convincing compared to different existing methods.**

Keywords*—Sentence Generation; Long Short-Term Memory; Word Embedding*

I. INTRODUCTION

Sentence generation means generating new words or sequences according to a specific context. This process can be considered as a subset of Natural Language Generation. Deep Learning is treated as a popular field in implementing this sort of concept. In sentence generation, preserving the context is strictly maintained. This preservation of context assures the process of maintaining both syntactic and semantic rules. Thus the resulting sequence secures the correct format as well as the relevant meaning of it. Sentence generation can be useful in text summarization, providing suggestions in the search section and also in producing predicted replies for an automated system or chatbot.

The proposed system is trained with the relevant data and the objective of the system is to understand the given input sequence and generate relevant new words. At first, all the unique words are split to form a vocabulary. All the words get tokenized and the corresponding n-grams are additionally generated. For acquiring uniformity in the lengths of different sequences, padding is applied. There are several layers in the sequential model of our system. The first layer is the embedding layer. In this layer, the tokens are converted into dense vector forms. There is a Long Short-Term Memory layer after that. This type of recurrent neural network is used as it is more effective to remember several states. Bidirectional LSTM [1] is capable of remembering both the future and previous states. Hence, a bidirectional LSTM layer is also used in the proposed model of the system.

The rest of the paper is structured as follows: Section II offers a concise background of the related research works. The proposed methodology is illustrated in Section III in great detail. Section IV exhibits the result analysis of the system. Then, the conclusions are outlined in Section V.

II. RELATED WORK

An immense amount of work has been done in sentence generation. In [2], Yadav et al. applied Corpus-based text generation. From input, they generated a bigram probability matrix and then lattice of bigrams. After that, they have used DFS to generate All possible candidate sequences. They declared that their model can provide a more favorable outcome by utilizing Part-of-Speech trigrams.

Bowman et al. [3] proposed a variational autoencoder based language model where words are represented using embedding vector. They investigated the latent space determined by their model and able to produce consistent and distinct sentences by continuous sampling. Segi et al. [4] introduced a template-based sentence-generation system. They recorded a relatively small sentence set and used dynamic programming for comparison and unification of formats. For handling optimization problems, they estimated the number of times that phrases and paths are included within a specific sentence set.

Song et al. [5] applied latent space expanded variational autoencoder (LSE-VAE) model to generate sentences from a continuous latent space. Their model can discover a huge informative latent space by arranging several prior latent configurations for distinct sentences and organizing the latent space corresponding to sentence similarity. For generating sentences, Krukaset et al. [6] employed two-sentence generation machines. A Manual method analyzes the sentences and discovers fixed patterns that are used by these machines. In their work, directed graphs are used to implement the fixed patterns.

For creating sentences, Jaya et al. [7] used the ontology technique which is utilized to render domain knowledge. They emphasized on generating sentences using specified grammar. They merged similar sentences for improving correctness. Their system focuses on the production of sentences for providing a particular story.

In this work, for the generation of sentences a model is proposed which uses word embedding and LSTM layers.

III. METHODOLOGY

The system is trained with a suitable dataset containing a large number of words. The unique words are used to form a vocabulary. Consequently, the labels and features are retrieved and Long Short-Term Memory architecture is used to generate new words according to the context. The overall

978-1-7281-7366-5/20/$31.00 ©2020 IEEE

Authorized licensed use limited to: Auckland University of Technology. Downloaded on December 23,2020 at 00:09:53 UTC from IEEE Xplore. Restrictions apply.

steps are divided into some major modules which are described below.

A. Data Collection and Description

We used a dataset on a news summary that can be found in [8]. The dataset consists of six attributes namely author, date, headlines, read_more, text, and ctext. From these attributes, the 'text' column is kept for further processing and all the other columns and null values are removed.

B. Processing Data for the Model

In this step, we process the collected data and prepare them to train the model. This module is divided into several sub-modules that are described here is detail.

1). Creating Vocabulary: First of all, we split every word from the data and gather all unique words to form the vocabulary.

2). Tokenization: Every unique word from the vocabulary gets tokenized, thus each word becomes a token and can be identified by particular number. Considering "**After the creation of the model, the accuracy is examined**." this sentence as an example. After tokenization phase this sentence becomes:

[2, 1, 3, 4, 1, 5, 1, 6, 7, 8]

Here, 2 represents the word '**After**', 1 represents '**the**', 3 represents '**creation**', 4 represents '**of**', 5 represents '**model**', 6 represents '**accuracy**', 7 represents '**is**', 8 represents '**examined**'. Here, a particular number is assigned to every unique word. Thus, every unique word act as a token.

3). N-grams Generation: We create corresponding n-grams of every sentence, starting from bigram up to length of the sentence. Some corresponding n-grams of the previously mentioned sequence are:

n-grams (for 2 tokens) = [2, 1]

n-grams (for 3 tokens) = [2, 1, 3]

n-grams (for 4 tokens) = [2, 1, 3, 4]

n-grams (for 5 tokens) = [2, 1, 3, 4, 1]

… … …. … … … … … … …

4). Padding: As all the generated n-grams are of different lengths, they are padded to acquire uniformity in the length. We pre-pad the sequences according to the biggest sequence length of the dataset. The pre-padded sequences of some n-grams determined in the previous phase are: (suppose the length of the longest sentence = 10)

[0, 0, 0, 0, 0, 0, 0, 0, 2, 1]

[0, 0, 0, 0, 0, 0, 0, 2, 1, 3]

[0, 0, 0, 0, 0, 0, 2, 1, 3, 4]

[0, 0, 0, 0, 0, 2, 1, 3, 4, 1]

5). Retrieving Labels and Features: In this step, we separate the last indexed value from each pre-padded sequence as the label, and all other values are stored as features. For example, for the padded sequence [0, 0, 0, 0, 0, 2, 1, 3, 4, 1] we consider [0, 0, 0, 0, 0, 2, 1, 3, 4] as features and [1] as the label. This is illustrated in Table I.

TABLE I. SEPARATED FEATURES AND LABELS

Features	Label
[0, 0, 0, 0, 0, 0, 0, 0, 2]	[1]
[0, 0, 0, 0, 0, 0, 0, 0, 2, 1]	[3]
[0, 0, 0, 0, 0, 0, 0, 2, 1, 3]	[4]
[0, 0, 0, 0, 0, 0, 2, 1, 3, 4]	[1]

TABLE II. LABEL AND RESPECTIVE ONE-HOT VECTOR

Label	One-hot Vector
[1]	[0, 1, 0, 0, 0, 0, 0, 0, 0, 0]
[3]	[0, 0, 0, 1, 0, 0, 0, 0, 0, 0]
[4]	[0, 0, 0, 0, 1, 0, 0, 0, 0, 0]
[1]	[0, 1, 0, 0, 0, 0, 0, 0, 0, 0]

6). One-hot Encoding: The separated labels are converted into one-hot vectors. This one-hot encoded form of the labels and the corresponding features are passed to the model. The one-hot encoded form of labels are shown in Table II where for illustration purpose, it is considered that the total number of classes = total number of unique words = 10.

C. The Proposed Model

Our proposed model has several layers inside it in a sequential manner. The proposed model is illustrated in Fig. 1. The different layers are described as follows.

1). Embedding Layer: Word embedding [9], [10], [11], [12] maps a set of words to vector form for improving the ability of neural networks. It is more efficient for neural networks to work on numerical data rather than text ones. This layer is used to convert the tokens or positive integers into vector form. Fig. 2 represents a word embedding for a specific position.

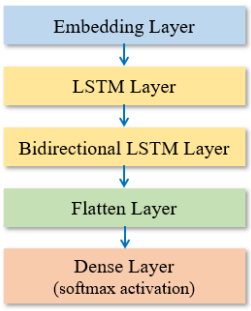


Fig. 1. The proposed model of the system

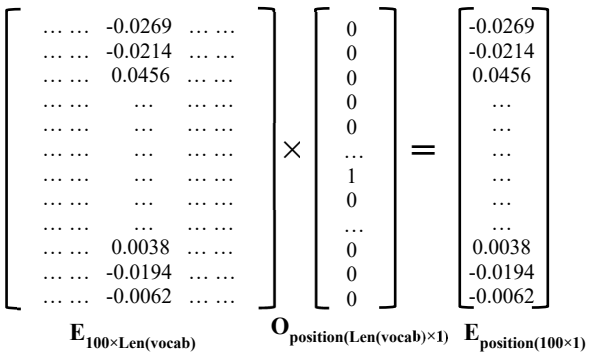


Fig. 2. Embedding for a word

Authorized licensed use limited to: Auckland University of Technology. Downloaded on December 23,2020 at 00:09:53 UTC from IEEE Xplore. Restrictions apply.

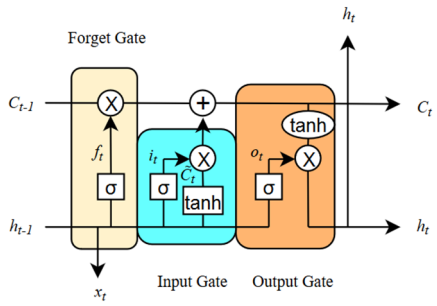


Fig. 3. Long Short-Term Memory layer

2). Long Short-Term Memory (LSTM) Layer: LSTM layer follows recurrent neural network architecture. The LSTM layer of our model has output space dimension equals to 100. LSTM network [13], [14], [15], [16], [17], [18], [19] has three gates. The architecture is shown in Fig. 3.

a). Input Gate: In this gate, sigmoid function determines which values of the input should be used to modify the memory.

$$i_t = \sigma(x_t U^i + h_{t-1} W^i) \quad (1)$$

$$\tilde{C}_t = tanh(x_t U^g + h_{t-1} W^g) \quad (2)$$

Here, i_t is called input gate, corresponding W *(i.e.* W^i*)* refers to the weight that connects to the recurrence layer at timestamp $t-1$ and to the hidden layer at timestamp t. Again, corresponding U *(i.e.* U^i*)* refers to the weight that connects to the hidden layer at timestamp $t-1$ and to the recursive layer at timestamp t. Also, σ represents the sigmoid function, h_{t-1} represents output of the previous LSTM block at timestamp *t-1* and $\tilde{C}_t$ represents candidate for cell state at timestamp t.

b). Forget Gate: Here sigmoid function determines what values to be discarded. In eq. (3). f_t represents the forget gate.

$$f_t = \sigma(x_t U^f + h_{t-1} W^f) \quad (3)$$

c). Output Gate: The output is decided here using input and the memory block. In eq. (4). o_t represents the forget gate.

$$o_t = \sigma(x_t U^o + h_{t-1} W^o) \quad (4)$$

$$C_t = \sigma(f_t * C_{t-1} + i_t * \tilde{C}_t) \quad (5)$$

$$h_t = tanh(C_t) * o_t \quad (6)$$

Here, the corresponding W *(i.e.* W^o*)* refers to the weight that connects to the recurrence layer at timestamp $t-1$ and to the hidden layer at timestamp t. Again, corresponding U *(i.e.* U^o*)* refers to the weight that connects to the hidden layer at timestamp $t-1$ and to the recursive layer at timestamp t. Also, σ represents the sigmoid function, h_{t-1} represents output of the previous LSTM block at timestamp $t-1$, h_t represents output of the previous LSTM block at timestamp t, C_t represents cell state at timestamp t and $\tilde{C}_t$ represents candidate for cell state at timestamp t.

3). Bidirectional LSTM Layer: Because of having two hidden layers in opposite directions, bidirectional LSTM [20] can work on both past and future states at a time.

$$h_t = \mathfrak{H}(W_{xh} x_t + W_{hh} h_{t-1} + b_h) \quad (7)$$

$$y_t = W_{hy} h_t + b_o \quad (8)$$

TABLE III. SUMMARY OF THE PROPOSED MODEL

Layers	Output Shape	Parameters
Embedding	(None, 46, 100)	214500
LSTM	(None, 46, 100)	80400
Bidirectional LSTM	(None, 200)	160800
Flatten	(None, 200)	0
Dense	(None, 2145)	431145
Total Trainable parameters		**886,845**

The corresponding W (*i.e.* W_{xh}, W_{hh} and W_{hy}) denotes the weight matrices, $\mathfrak{H}$ refers to the hidden layer of the bidirectional LSTM architecture, h_{t-1} represents output of the previous LSTM block at timestamp $t-1$, h_t represents output of the previous LSTM block at timestamp t. The bidirectional LSTM layer of our model has output dimension same as the conventional LSTM layer that means 100.

4). Flatten Layer: This layer takes the output of the previous layer and puts the value in a single vector i.e. this layer flattens the data and feeds it into a fully connected layer.

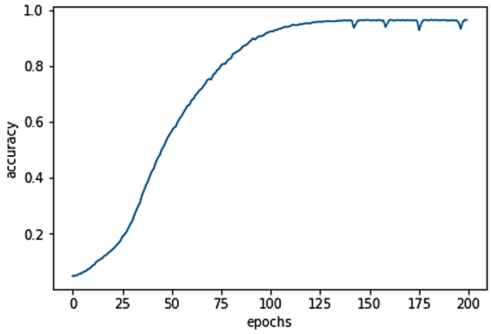
5). Dense Layer: A dense layer is also applied in our model with a non-linear activation function called 'softmax' and dimensionality of the outer space equals to the length of the vocabulary. Now, our model is compiled with 'adam' optimizer and fit with features and labels. The summary of the proposed model is shown in Table III.

D. Generating New Word

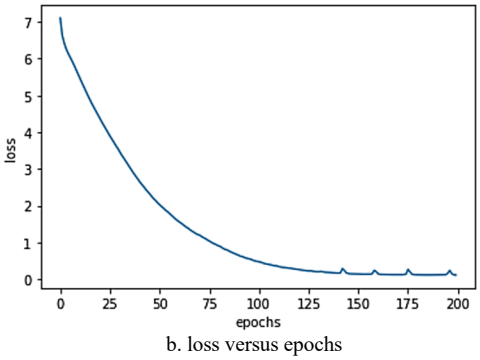
After training the model, an input is given to it in tokenized form. The model predicts a new word after each iteration maintaining the context. After generating every new word, the word also is added to the previous input and the new combination is considered as the next input.

IV. RESULT ANALYSIS

The training set of the system includes 80 percent of the dataset. After fitting the training data into the model, the respective loss and accuracy rate is observed. It is visible from



a. accuracy versus epochs



b. loss versus epochs

Fig 4. Accuracy and Loss curve for 200 epochs for the proposed system

Authorized licensed use limited to: Auckland University of Technology. Downloaded on December 23,2020 at 00:09:53 UTC from IEEE Xplore. Restrictions apply.

TABLE IV. COMPARISON WITH EXISTING METHODS

Methods	Input	Output
J. Brownlee [21]	be no mistake about it: it was neither more nor less than a pig, and she felt that it would be quit	be no mistake about it: it was neither more nor less than a pig, and she felt that it would be quit e aelin that she was a little want oe toiet ano a grtpersent to the tas … … …
I. Danish [22]	describe the problem	describe the problem please attend to
The Proposed Method	be no mistake about it: it was neither more nor less than a pig, and she felt that it would be quit	be no mistake about it: it was neither more nor less than a pig, and she felt that it would be quit maharashtra police medical a her they has it social it been which was chopping the revoked … … …
	describe the problem	describe the problem alleged by them
	the Dhaka high court reduced the compensation award	the Dhaka high court reduced the compensation award said office turnaround administration women

Fig 4. that the rate of loss is gradually decreasing and the rate of accuracy per epoch is gradually increasing. The rates are plotted into distinct graphs to visualize the effect. According to Table IV, the first compared model produces words which don't even exist. The meaning of the result of that model also isn't too convenient. For the same input sequence, our proposed model generates more meaningful result. Our model doesn't produce any non-existing word as well. In the second comparison, the compared model produces a nonsensical sequence. For the same input sequence, the result generated by our model makes perfect sense. We also portray a third example generated by the proposed system so that the robustness of the system can be more comprehensible.

V. CONCLUSION AND FUTURE WORK

This paper demonstrates the idea of building a model that can generate new sentences using a deep learning approach. Our techniques are utterly based on the idea of word embedding and Long Short-Term Memory (LSTM) architecture which is a modified version of Recurrent Neural Networks (RNN). As it is more convenient for the neural network to work with numerical data, the text data are converted into vector form using word embedding. For maintaining the proper context LSTM and Bidirectional LSTM network are applied. Bidirectional LSTM is a very important layer for the model as it can operate both present and past state at a single time. We trained the model successfully and the results are thoroughly analyzed and compared with different models as well.

In future work, we wish to further enhance the model so that it can be used as a text summarizer or a dynamic suggestion provider for a chatbot.

REFERENCES

[1] Z. Huang, W. Xu, and K. Yu, "Bidirectional lstm-crf models for sequence tagging," *arXiv preprint arXiv:1508.01991*, 2015.

[2] A. K. Yadav and S. K. Borgohain, "Sentence generation from a bag of words using n-gram model," in *2014 IEEE International Conference on Advanced Communications, Control and Computing Technologies.* IEEE, 2014, pp. 1771–1776.

[3] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio, "Generating sentences from a continuous space," *arXiv preprint arXiv:1511.06349*, 2015.

[4] H. Segi, R. Takou, N. Seiyama, T. Takagi, H. Saito, and S. Ozawa,"Template-based methods for sentence generation and speech synthesis," in *2011 IEEE International Conference on Acoustics, Speech and Signal Pro-cessing (ICASSP).* IEEE, 2011, pp. 1757–1760.

[5] T. Song, J. Sun, B. Chen, W. Peng, and J. Song, "Latent space expanded variational autoencoder for sentence generation," *IEEE Access*, vol. 7, pp. 144 618–144 627, 2019.

[6] W. Krukaset, N. Krukaset, and C. Khancome, "Thai sentence generation machine employing fixed patterns," *in 2017 IEEE 19th International Con-ference on High Performance Computing and Communications Workshops (HPCCWS)*. IEEE, 2017, pp. 70–73.

[7] A. Jaya and G. Uma, "A novel approach for construction of sentences for automatic story generation using ontology," *in 2008 International Conference on Computing, Communication and Networking*. IEEE, 2008, pp. 1–4.

[8] sunnysai12345 "sunnysai12345/News_Summary." github.com. https://github.com/sunnysai12345/News_Summary (accessed Mar. 15, 2019).

[9] O. Levy and Y. Goldberg, "Neural word embedding as implicit matrix factorization," in *Advances in neural information processing systems*, 2014, pp. 2177–2185.

[10] X. Rong, "word2vec parameter learning explained," *arXiv preprint arXiv:1411.2738*, 2014.

[11] D. Karani, "Introduction to Word Embedding and Word2Vec." towardsdatascience.com. https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa (accessed Feb. 26, 2020).

[12] S. Agrawal, "What the heck is Word Embedding." towardsdatascience.com. https://towardsdatascience.com/what-the-heck-is-word-embedding-b30f67f01c81 (accessed Aug. 23, 2019).

[13] M. Farahani, M. Farahani, M. Manthouri, and O. Kaynak, "Short-Term Traffic Flow Prediction Using Variational LSTM Networks," *arXiv e-prints*, p. arXiv:2002.07922, Feb 2020.

[14] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm," 1999.

[15] A. Graves, "Generating sequences with recurrent neural networks," *arXivpreprint arXiv:1308.0850*, 2013.

[16] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[17] K. Yao, T. Cohn, K. Vylomova, K. Duh, and C. Dyer, "Depth-gated lstm," *arXiv preprint arXiv:1508.03790*, 2015.

[18] A. Mittal, "Understanding RNN and LSTM." towardsdatascience.com. https://towardsdatascience.com/understanding-rnn-and-lstm-f7cdf6dfc14e (accessed Oct. 12, 2019).

[19] AlindGupta, "Text Generation using Recurrent Long Short Term Memory Network." geeksforgeeks.org. https://www.geeksforgeeks.org/text-generation-using-recurrent-long-short-term-memory-network/ (accessed Nov. 17, 2019).

[20] Z. Yu, V. Ramanarayanan, D. Suendermann-Oeft, X. Wang, K. Zechner, L. Chen, J. Tao, A. Ivanou, and Y. Qian, "Using bidirectional lstm recurrent neural networks to learn high-level abstractions of sequential featuresfor automated scoring of non-native spontaneous speech," in *2015 IEEEWorkshop on Automatic Speech Recognition and Understanding (ASRU)*. IEEE, 2015, pp. 338–345.

[21] J. Brownlee, "Text Generation With LSTM Recurrent Neural Networks in Python with Keras." machinelearningmastery.com. https://machinelearningmastery.com/text-generation-lstm-recurrent-neural-networks-python-keras/ (accessed Dec. 26, 2019).

[22] I. Danish, "Sentence Prediction Using a Word-level LSTM Text Generator — Language Modeling Using RNN." medium.com. https://medium.com/towards-artificial-intelligence/sentence-prediction-using-word-level-lstm-text-generator-language-modeling-using-rnn-a80c4cda5b40 (accessed Dec. 26, 2019).

Authorized licensed use limited to: Auckland University of Technology. Downloaded on December 23,2020 at 00:09:53 UTC from IEEE Xplore. Restrictions apply.