International Journal of Multidisciplinary Research and Growth Evaluation
ISSN: 2582-7138
Received: 06-08-2021; Accepted: 23-08-2021
www.allmultidisciplinaryjournal.com
Volume 2; Issue 5; September-October 2021; Page No. 208-211

A simple method for sentences generation based on deep learning

Giao N Pham
Department of Computing Fundamentals, FPT University, Hanoi, Vietnam

Corresponding Author: **Giao N Pham**

Abstract
Long short-term memory (LSTM) units on sequence-based models are being used in translation, question-answering systems, and classification tasks due to their capability of learning long-term dependencies. Currently, LSTM networks are providing impressive results on text generation models by learning language models with grammatically stable syntaxes. In this paper, I would like to present a simple method to generate text using LSTM networks. Experimental results showed that the LSTM method is suitable for text generation with low loss in the training/testing processes.

Keywords: Text generation, LSTM networks, Vectorization, Recurrent Neural Networks

1. Introduction
Through the use of Long Short-Term Memory Networks, I aim to model the English language as closely as possible, such that model created is able to generate coherent sequences of words, namely, sentences. To achieve this, every word is transformed into a feature vector, which is used as input. Its corresponding output is the next most probable word to follow. The network will learn these probabilities through the use of a large list of human-generated sentences, more specifically a literature novel. To clarify, my paper is organized as follow: The related work is discussed in Sec. 2; the proposed method and experimental results is described in detail in Sec. 3 and Sec. 4; and finally conclusion is shown in Sec. 5.

2. Related Works
Recurrent Neural Networks (RNNs) are applied to the analysis of sequential information [1]. As opposed to other kinds of neural networks that handle each input and output independently, an RNN allows older information to linger. This concept is described in Fig. 1. Given a sequence of data, an RNN is able to predict the output(s) to follow by taking it's output back. This way, the network can compute a hypothesis h based on previous collected knowledge. This can be a very powerful characteristic. Unfortunately, RNNs do have some pitfalls. One of them is the problem of Vanishing Gradients. As the data moves along different neural networks, it also passes through many multiplication stages; this is necessary in order for the model to establish a connection between an output and data seen several steps before [1]. In this process, it's possible that gradients become too small for a machine to work with.

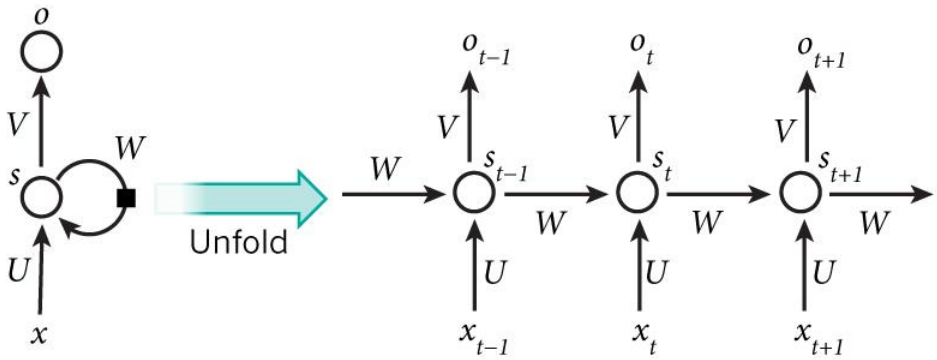

Fig 1: Graphic Representation of a Recurrent Neural Network.

Long Short-Term Memory (LSTM) Networks [2] handle the hidden state slightly differently than regular RNNs, but are also capable of learning long-term links. The repeating modules (namely, the memory cell) now has a different structure: Through a set of internal neural networks and several gates, this element is able to determine what information to store, and what to forget. This data flow can be visualized in Fig. 2.

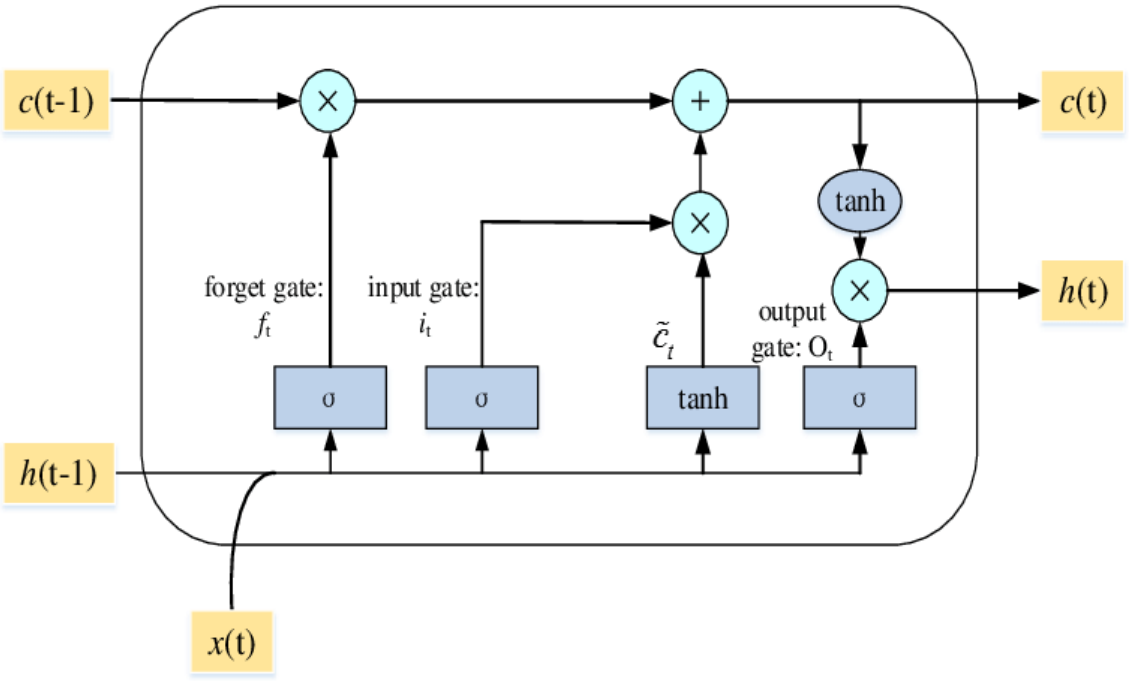

Fig 2: Long Short-Term Memory Cell Structure.

3. The Proposed Method

I created a deep learning system for generating text based on a given input as shown in Fig. 3. For this, I wanted to use the LSTM model- a deep learning multilayer recursive model [3], which can generate similar output based on a given input. The proposed method is as follows:

- Parser: converts the given text character into binary vectors.
- LSTM: the learning component- based on a given weight vector, and an input, can generate an output in the form of a binary vector.
- Weight vector: the brains behind the LSTM learning operation- the most time consuming part of the algorithm is to generate the weight vector. Once it's generated though, it's possible to save it to file and reuse it as many time as we'd like.
- Parser: Converts the binary vector back to text.

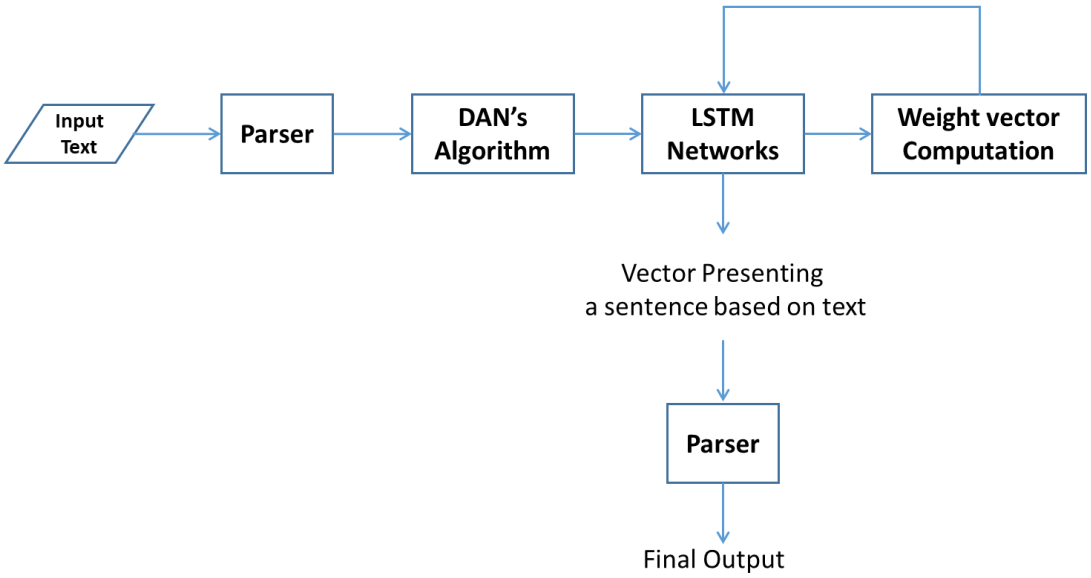

Fig 3: The Proposed Method.

4. Experimental Results

I used a maximum length of 15 per sentence and a sliding window of 3 when iterating through the original text. Following this idea, a total of 33, 475 patterns were created; this implies that there were some overlapping words. All sentences, or patterns, are also accompanied by their

corresponding target, which is the actual word to follow. For training, I used 100 epochs and only proceeded to save the weights if the loss improved for that specific epoch iteration.

In terms of cross entropy loss throughout training, the results are presented in Fig. 4. The loss, as expected, decreases over time, but stabilizing after a certain number of iterations (50).

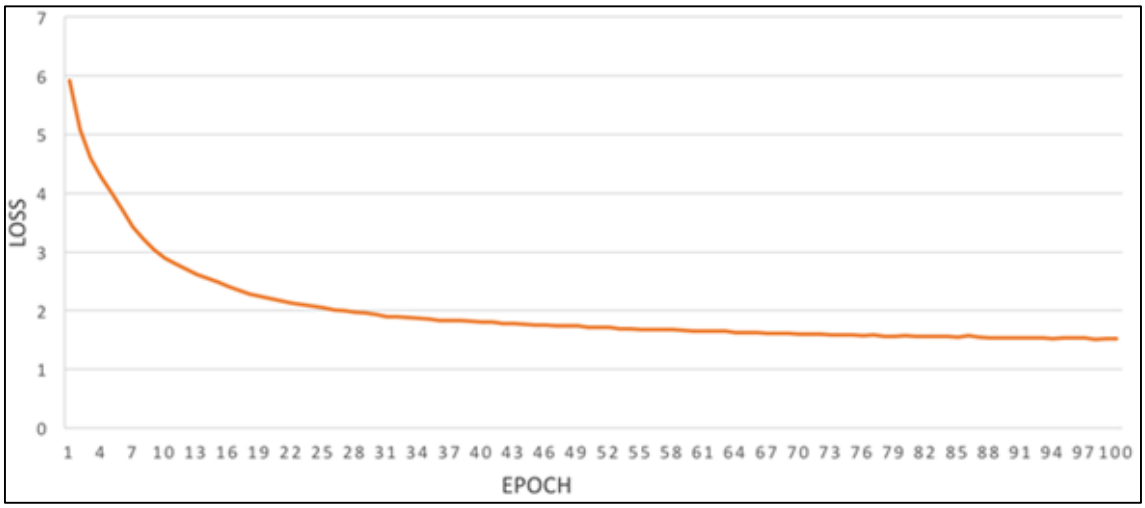

Fig 4: Results for Cross Entropy Loss over Epoch Iterations for the Base LSTM Model.

A number of possible additions to the model were considered in order to improve based on the LSTM networks. The first approach to improve the current model will be by varying the length of each sequence in the training set. I changed the length of each pattern from 15 to 5, and adjusted the sliding window accordingly, from 3 to 2. With this change, the number of training sets increased significantly from 33, 475 to 50, 218. The results for this experiment are shown in Fig. 5. Though the loss decreases relatively quickly, it's lowest only reaches 1.881628014 at epoch 98. For this reason, the length of 15 will be kept for future modifications. Moreover, it was the case for both outcomes that the loss did not decrease by much after epoch 50. Due to this, next experiments will be carried out using 50 epochs.

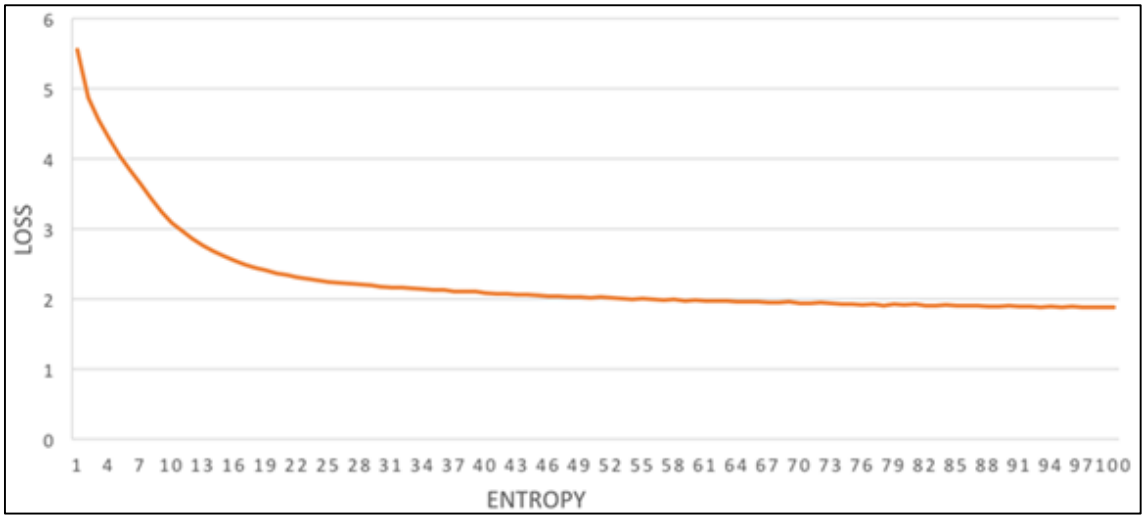

Fig 5: Results for Cross Entropy Loss over Epoch Iterations.

The second approach was to add a second LSTM layer with 256 memory units as well. Following this layer, a second Drop Out Layer with a drop-out rate of 20% was also added to accompany it. Experimental result is shown in Fig. 6, that is not improved. Even though it is decreased slightly after epoch 8, it wasn't much. The loss was essentially constant throughout the iterations; there wasn't much change from the value it started with to its lowest: 6.354732551. This may be too complex of a model for this kind of problem; more than one layer is likely not needed.

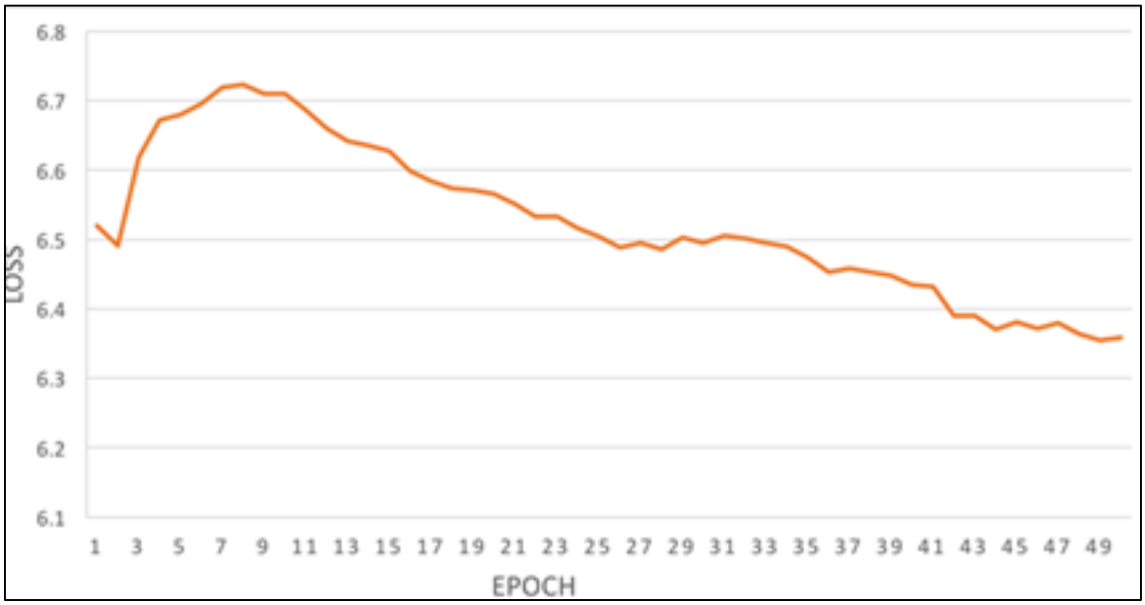

Fig 6: Results for Cross Entropy Loss over Epoch Iterations with a second LSTM Layer and Drop Out Layer.

5. Conclusion

In this paper, I presented and experimented text generation using the LSTM networks. Unfortunately, the experimental results of the proposed method were not satisfactory. The initial model created outperformed all other approaches for potential improvements. It's important to note that a number of these parameters may work differently with other training sets. For future work, I would like to attempt these methods once again with different data.

6. Acknowledgement

This work was supported by FPT University, Hanoi, Vietnam.

7. References

1. A Guide to RNN: Understanding Recurrent Neural Networks and LSTM Networks. Available online: https://builtin.com/data-science/recurrent-neural-networks-and-lstm, accessed on 11/09/2021.
2. A Gentle Introduction to Long Short-Term Memory Networks by the Experts. Available online: https://machinelearningmastery.com/gentle-introduction-long-short-term-memory-networks-experts/, accessed on 11/09/2021.
3. Text Generation with LSTM Recurrent Neural Networks in Python with Keras. Available online: https://machinelearningmastery.com/text-generation-lstm-recurrent-neural-networks-python-keras/, accessed on 11/08/2021.