




Empirical Laws of Natural Language Processing for Neural Language Generated Text

Sumedha^(✉)  and Rajesh Rohilla

Department of Electronics and Communications, Delhi Technological University, Delhi, India
rajesh@dce.ac.in

Abstract. In the domain of Natural Language Generation and Processing, a lot of work is being done for text generation. As the machines become able to understand the text and language, it leads to a significant reduction in human involvement. Many sequence models show great work in generating human like text, but the amount of research work done to check the extent up to which their results match the man-made texts are limited in number. In this paper, the text is generated using Long Short Term Memory networks (LSTMs) and Generative Pretrained Transformer-2 (GPT-2). The text by neural language models based on LSTMs and GPT-2 follows Zipf's law and Heap's law, two statistical representations followed by every natural language generated text. One of the main findings is about the influence of parameter Temperature on the text produced. The LSTM generated text improves as the value of Temperature increases. The comparison between GPT-2 and LSTM generated text also shows that text generated using GPT-2 is more similar to natural text than that generated by LSTMs.

Keywords: Long Short Term Memory networks (LSTMs) · Transformers · Generative Pretrained Transformer 2 (GPT-2) · Text Generation · Zipf's Law · Heap's Law

1 Introduction

Language is a very important part of our lives since it is something we use to communicate our thoughts. At an abstract level, language can be explained as a collection of alphabets and characters which can be accessed through some rules known as grammar. Even for humans, it takes years to learn a language because of its complexity. As the rules are not standard in every language, the existence of sarcasm and context adds extra fuzziness to the process of understanding it by the system. For example, the word duck in terms of cricket and general terms hold completely different meanings [10]. Due to all these challenges, natural language generation and processing have for quite some time been considered as among the most testing computational assignments.

Standard Neural Networks are not used for text generation because the length of input and output is not fixed, it may vary with each sentence. For this sequential data Recurrent Neural Networks (RNNs), Gated Recurrent Units (GRUs), Long Short Term Memory Networks (LSTMs), Encoder-Decoder Models, or some type of Transformer is

used. Since these are predictive models as well as generative models i.e., they can learn from sequence and generate a completely new sequence based on training data fed to it [10]. In this paper, we have generated text using LSTMs and Generative Pretrained Transformer 2 (GPT-2). This process consists of two parts first being preparing the data. In the preparation step various processes such as data cleaning and exploration, tokenization, character to integer mapping, applying sliding window technique, etc. are done, so that data can be fed to the machine learning model. The next step is training the model which includes defining the type of model, adding embedding, neural network and dropout layers, deciding values and types of various hyper-parameters for the model, etc.

After generating text, we have evaluated how close the neural language generated text is to human-generated text by checking if it follows Zipf's law and Heap's law, two empirical laws followed by every man-made text in every language. Along with this, we have seen the dependence of text generated on a parameter called Temperature and compared text generated by LSTMs to that generated using GPT-2.

2 Related Work

The task of text being generated by a machine was first seen in mid the 1960s with the emergence of ELIZA, a chatbot-like computer program built at MIT Lab for Artificial Intelligence. ELIZA breaks the input into sentences, parses it based on a simple pattern, and searches for keywords. Based on that keyword it generates a generalized response and if no keyword is found it asks the user to elaborate the input [6]. Although it works well, the amount of intelligence involved is very little.

Comparatively more commercial and intelligent text generation systems using Markov Chains first appeared in March 2017. Markov Chains predict the next word using the current word, as the output depends only on the current word, text loses semantics and context [11]. To overcome this problem, using Recurrent Neural Networks (RNNs) is recommended. In RNNs the current output is a function of present input as well as output of the previous neural network, hence it recollects the past and, conclusions it makes are also influenced by what it has learned from the past as well as current data being fed to it. As we go on with training RNNs, weights try to adjust themselves to minimize the error. In this process weights at the end will have more influence on the text generated as compared to weights at beginning of the network and the weights at the start slowly become zero. This process is known as vanishing gradients and is solved by using Long Short Term Memory networks (LSTMs) [10]. LSTMs are an exceptional sort of RNNs, equipped for learning comprehensive dependencies. They have an extra input known as cell state. The cell state is updated in each step such that if weights are too high it lowers them, if they are much less it increases them, hence avoiding the problem of vanishing or exploding gradients [12]. Along with research in the field of text generation using LSTMs, possibilities of using Generative Adversarial Networks (GANs) for text generation are also being explored [5].

The most recent and well-performing machine learning models for text generation are Transformers. These models work on self-attention mechanism and are developed using encoders and decoders. All the best performing neural network architectures in

the field of Natural Language Processing are found to be variants of transformers e.g.: BERT, GPT-2, etc., and the recent researches are focused on improving these models [9].

3 Methods

3.1 Zipf's Law and Heap's Law

All human-generated texts in all the languages follow Zipf's law and Heap's law. If our machine-generated text also follows these laws, we say that this neural language generated text is close to the natural language generated text.

According to Zipf's law, for all human-generated texts, the rank-frequency distribution is an inverse relation (frequency is proportional to the inverse of rank) [4]. In simple terms it can be explained as, the word that occurs the greatest number of times occurs two times more than the word that occurs the second most number of times, three times more than the third most frequently occurring word, and so on [11]. In natural language processing, Zipf's law is mostly used in text compression algorithms, so that we know about the frequency of words and algorithm compresses words that occur very frequently and not the rarely occurring ones. Along with text compressors it is also used in various NLP Algorithms and text generators [12]. Although it is an empirical law, many explanations have been attributed to its occurrence such as the Principle of least effort which says that, people tend to read and write easily so they use the same words more and more and it also gets passed to future generations.

Mathematically, Zipf's law can be represented as:

$$f(r) = Cr^{-k} \quad (1)$$

where $f(r)$ is the frequency of term, r is its rank, C is a constant and k has value approximately equal to 1.

Heap's law states that as the size of document increases, the rate at which the number of distinct words increase in it takes a downturn e.g.: Suppose in a document with 1000 words no. of unique words are 100, then for a document with 2000 words no. of unique words will be less than 200, for a document with 3000 words no. of unique words will be much less than 300 and so on [8].

The archived meaning of Heaps' law says that the quantity of exceptional words in content of n words is approximated by:

$$Q(n) = An^k \quad (2)$$

where A is a positive constant which is usually up to 100, k is between 0 and 1, most frequently in range the 0.4 to 0.6 [11].

3.2 Long Short Term Memory Networks

Recurrent Neural Networks (RNNs) are the neurons which receive input from the previous cell as well as present cell. This can be seen in Fig. 1, where $y_2(t) =$

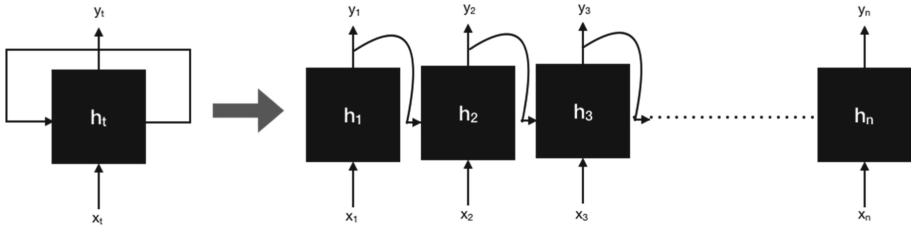


Fig. 1. Unrolled RNN cell

$f(x_2(t) + y_1(t - 1))$. Cells that are the function of input from previous cell are known as memory cells. As a result, RNNs remember the past. These are designed to work with sequential data like sentences, audio, video, etc. [10].

While training the network on large sequences RNNs begin to forget the starting part due to vanishing gradient issue which is solved utilizing LSTMs. Remembering details for a long time comes naturally to LSTMs because of their structure and design.

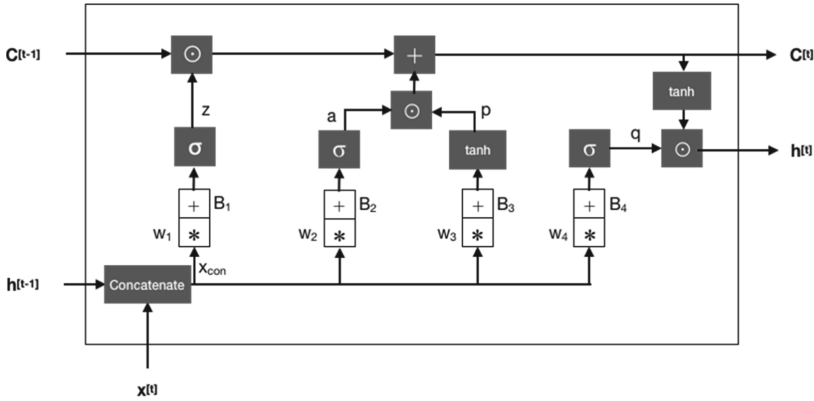


Fig. 2. LSTM cell

In Fig. 2, $C^{[t]}$ represents the memory cell i.e., it provides the memory feature to LSTM cells, $h^{[t-1]}$ is the output of the previous cell. The LSTM generates $h^{[t]}$ and $c^{[t]}$ as its outputs [13].

Its functioning can be explained in four steps or layers:

- First, a forget Layer makes a decision on information to be kept and thrown away. In this step, we pass $h^{[t-1]}$ and $x^{[t]}$ into a sigmoid function, which generates an output between 0 and 1. Output 0 represents forgetting that information and an output of 1 represents keeping that information. e.g.: When working with a subject we may need to remember its gender while selecting that pronoun but may not need to remember that feature for the next subjects. Hence, this information can be forgotten.

$$z = \sigma(w_1 \cdot x_{con} + B_1) \quad (3)$$

- In the second step, we decide about the new information to be stored in cell state $c^{[t]}$. For this, a sigmoid layer decides values that are required to be updated. After this, a tanh layer (as it distributes gradients, hence prevents vanishing/exploding) gives an array of new values.

$$a = \sigma(w_2 \cdot x_{\text{con}} + B_2) \quad (4)$$

$$p = \tanh(w_3 \cdot x_{\text{con}} + B_3) \quad (5)$$

$$q = \sigma(w_4 \cdot x_{\text{con}} + B_4) \quad (6)$$

- In this step we update the old state cell by deciding the output for $c^{[t]}$ and discarding information about old subject and adding new details.

$$C^{[t]} = z \odot C^{[t-1]} + p \odot a \quad (7)$$

- For the last step, we update the output $H(t)$. First, a sigmoid layer is present which tells about the part of the cell state which will contribute to $h[t]$. Then, its result is given to the tanh function and is multiplied by sigmoid gate's output.

$$h^{[t]} = q \odot \tanh(C^{[t]}) \quad (8)$$

3.3 Transformers

Transformers are a new family of Machine Learning models introduced in 2017. These work on the self-attention model, that is we look for a relationship between different input sequences [2]. This means they don't remember the whole sentence at once, but a perimeter α is assigned. $\alpha_{<1,1>}$ decides how much value the first network holds while generating the first word, similarly $\alpha_{<2,1>}$ decides how much value the first network holds while generating the second word and so on. In transformers, this process is done multiple times, so this is known as multi-headed self-attention [3].

As, can be seen in Fig. 3, a transformer cell consists of encoder and decoders blocks. An encoder block is further a collection of many encoders connected serially, similarly, a decoder block is also a collection of connected decoders. An encoder converts text to word embeddings which are fed to the decoder, which as output generates text. Many latest models based on the transformer architecture are BERT, GPT-2, GPT-3, etc. In this paper, text is generated using Generative Pretrained Transformer 2 (GPT-2), which is based on a transformer model and uses decoder blocks. These have shown remarkable performance in every field of Natural Language Processing including text generation [9].

4 Experiments and Results

4.1 Dataset Used

As dataset, the famous book "The Adventures of Sherlock Holmes" by Sir Arthur Conan Doyle is utilized. The book is made available through Project Gutenberg and contains

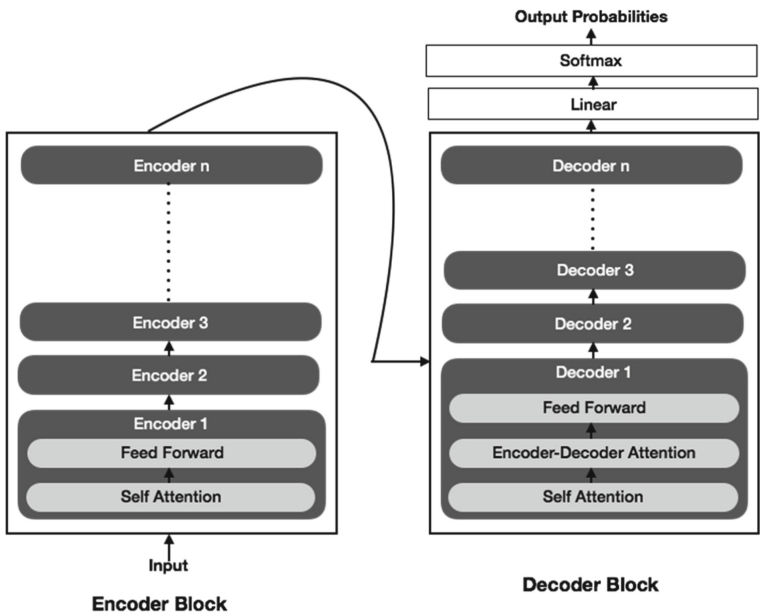


Fig. 3. Transformer cell

a total of 594,197 characters. Most ordinarily happening words in the dataset and their recurrence can be seen in Fig. 4.

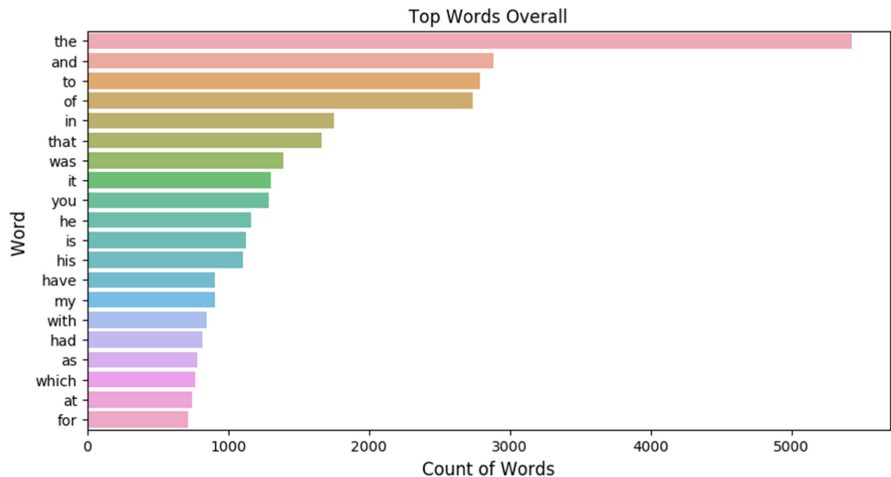


Fig. 4. Most ordinarily happening words in the dataset and their recurrence

4.2 Approach

- Explored the dataset to find the total number of characters, most commonly used words, etc.
- Cleaned the data and pre-processed it by applying Tokenization and removing punctuations and stop words
- Prepared the data for training by applying Sliding Window Technique and Character to Integer Mapping
- Tuned hyperparameters and finally generated sequential model with the accompanying highlights:

Number of Layers: LSTM = 2, Dropout = 1, Dense = 2

Loss computed: Categorical cross-entropy

Optimizer employed: Adam

Activation function: ReLU

Input Sequence Length: 25 words

Batch Size: 192

Count of neurons in each hidden layer: 250

- Created LSTM based model, fitted it, and generated text for different values of word length and hyper-parameter Temperature in range 0.1 to 3
- Generated text using GPT-2 for different word lengths
- Compared how well LSTM and GPT-2 generated text follows Zipf's law and Heap's law
- Compared the texts generated using LSTMs and GPT-2

4.3 Defining the Terms Used

4.3.1 Tokenization

It is a process of expressing the words in a way that they can be processed by the system and is usually the first step while processing the text data. In the process of tokenization, we divide the original text into smaller pieces. The units thus generated are called tokens. Tokens can be words or characters depending upon the type used.

4.3.2 Sliding Window Technique

After obtaining an array of words as tokens, we prepare the model for training. First step in it being the sliding window technique. In this process a window of fixed size (sequence length is taken to be 25 words for this project) is operated on some text, then that text is learned. After learning that window is moved one token at a time and remembers that. This process continues till we reach end of the sequence.

4.3.3 Character to Integer Mapping

In this process, each character is assigned a unique integer. It is done because for machines it is easier to read numbers as compared to text data. This process reduces complexity and increases speed of the model [8].

4.4 Results Obtained

4.4.1 Variation of Text Generated Using LSTMs with Parameter Temperature

Table 1. Text generated using LSTMs keeping document length = 50 words and varying Temperature in range 0.1 to 3

(Temperature, document length)	Text generated
(0.1, 50)	Each when we followed the back from some mark the openly preceded in spite of the slight man in the man work and the heading gave the north taken and dealings respectable 4 and of an attacked of the drug out of the help and with a small public face
(0.5, 50)	Is used living and violates support in the dead of reading the work seen of which was whine beside the edge of the great house which had been hereditary sense hands across to its crop to explain you through the first hand of ballarat in the fancier faced hat which
(1, 50)	All machinery is given you through hand where is very such so long that miss remark is its laughed or coupled' judgment from all the house was the disposition of the city branch of it and save the full which he had promised to have spoken to us to
(1.5, 50)	Across together the show little gloom of burning in the u.s. and accordance heard repeated sign brown and distributed project gutenbergm electronic works and you do not narrow to be
(2, 50)	Robberies three of a shriek in disappearing acid with the inspector tragedy of the spellbound sun were inches of hercules by the manager of those gaiters was of the same residence the light men and the help like being waistcoat yet in the aperture his is was as his sleeping
(2.5, 50)	Behind the white curling edge which are thrown at a manner bow on peeped on a shoulders face which is over the house a show plain man hair crystallised the smaller observe of the front of his lantern and left the date of the sea indexing i do not touch
(3, 50)	The edge there was breaches crushed and violates support in a poisoning this and door to the same corridor by the odessa vault of water which trafalgar him to come investment we had just a very asked witted carried under the irene world against me by order to prevent the

Temperature is a hyperparameter in neural networks which controls the randomness of text generated or any output by scaling the un-normalized predictions before applying activation function.

In Table 1, we have measured the effect of variation of hyperparameter Temperature on the text generated, by keeping the word length in document as constant and we can see that as temperature (randomness) increases grammatical errors decrease and sentences start making comparatively more sense.

4.4.2 Text Generated Using GPT-2 for Different Word Lengths

We implemented a pre-trained transformer model using Hugging Face and generated a text of lengths = 25, 50, 75 and 100 with random seed text [10]. The results obtained can be seen from Figs. 5, 6, 7 and 8.

```
0: Watson you are in an accident?
Pam: Oh yes, and I would tell you, but I'm
```

Fig. 5. Text generated using GPT-2 for word length = 25

```
1: Watson you are the greatest football fan ever, what can you say in your defence?
"I could probably give you one reason why I love being a fan of rugby union so much.
"I'm not going to pretend to
```

Fig. 6. Text generated using GPT-2 for word length = 50

```
2: Watson you are no longer allowed to leave your job but your job does not matter. He tells me that his only aim is
to get people to know and respect his intelligence; this is important for a successful company, otherwise he doesn't
understand his customers. You are a smart man and I applaud your intelligence, but the rest is a complete waste of th
e time, energy
```

Fig. 7. Text generated using GPT-2 for word length = 75

```
3: Watson you are very well. Now you are saying that you would not get away with not getting away with the stuff. Are
you really implying that you were in a position where you knew that you were not making it a crime to say "fuck you"
in public to the police officer? Is that what you are implying?
Hmmm. Is that what you are implying? Is that what you are implying? It certainly strikes me as a strong indication of
some sort of crime of verbal
```

Fig. 8. Text generated using GPT-2 for word length = 100

From the obtained text we can see that the text generated by GPT-2 is understandable to great extent, but sudden change in topic can be observed for each word length. Comparing this to LSTMs generated text we can say that text by GPT-2 makes more sense and is grammatically better. Also, the process of text generation by GPT-2 is more computationally expensive than the process involving LSTMs.

Table 2. Word rank vs. frequency plot and number of unique words for text generated using LSTMs for different values of Temperature and Document Length in words

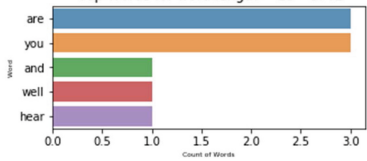
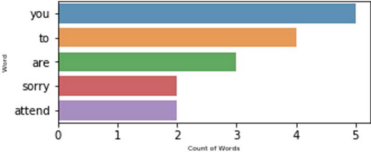
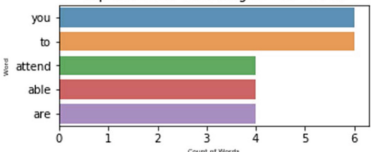
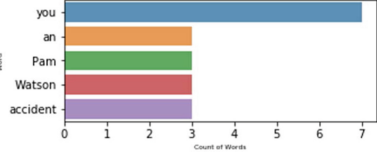
(Document Length in words, Temperature)	Word rank vs. frequency	Number of unique words
(25, 0.1)	<div><div>Top Words in (25,.1)</div></div>	21
(75, 0.1)	<div><div>Top Words in (75,.1)</div></div>	59
(25, 1)	<div><div>Top Words in (25,1)</div></div>	21
(75, 1)	<div><div>Top Words in (75,1)</div></div>	60
(25, 2)	<div><div>Top Words in (25,2)</div></div>	23
(75, 2)	<div><div>Top Words in (75,2)</div></div>	57

4.4.3 Verifying Empirical Laws for Text Generated Using LSTMs and GPT-2

From Table 2, we can see that for all temperatures and document lengths word rank and frequency are inversely proportional to each other.

For verifying these two power laws for GPT-2 we generated text with various word lengths and calculated number of unique words and word rank vs its frequency distribution for each document length and got the results as displayed in the following table.

Table 3. Word rank vs. frequency plot and number of unique words for the text generated using GPT-2 for different values of Document Length in words

Document Length in words	Word rank vs. frequency	Number of unique words
25	<div><p>Top Words for word length = 25 - GPT2</p></div>	18
50	<div><p>Top Words for word length = 50 - GPT2</p></div>	30
75	<div><p>Top Words for word length = 75 - GPT2</p></div>	35
100	<div><p>Top Words for word length = 100 - GPT2</p></div>	51

From Table 3, we can see that for all document lengths word rank and frequency are inversely proportional to each other for GPT-2 generated text as well. Hence, we can say that Zipf’s law is being followed by text generated by LSTMs and GPT-2.

On plotting Document length vs. Number of unique words for LSTMs and GPT-2 we get the plot as shown below:

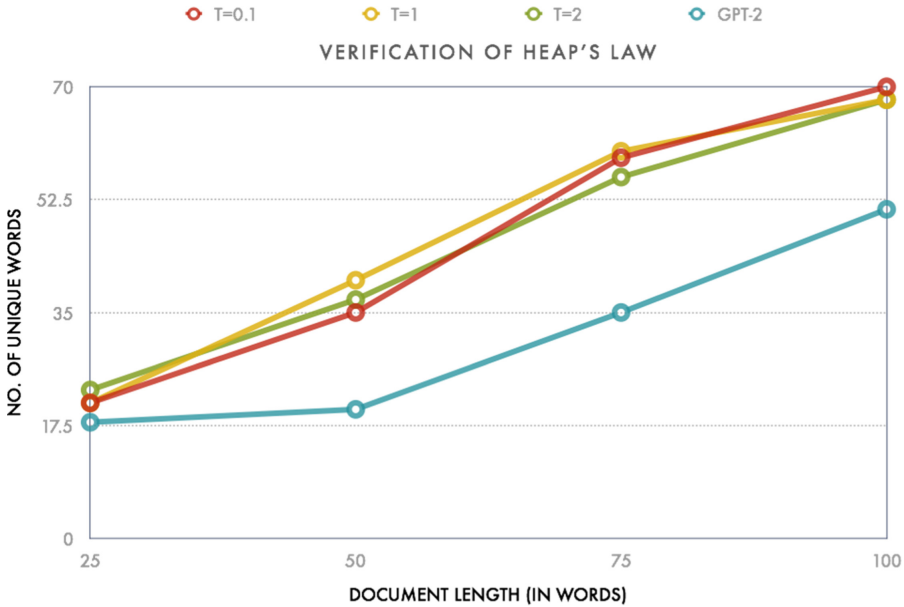


Fig. 9. Verification of Heap’s Law

From Fig. 9, we can see that as document size is increasing, the rate at which the number of distinct words is increasing has dropped. This tells us that Heap’s law is also being followed for the text generated by the neural language models (both LSTM networks and GPT-2).

4.5 Comparison with Existing Works

M. Lippi, M. A. Montemurro [1] generated text with LSTMs using torch-rnn package, with 2 LSTM layers and each layer consisting of 1024 cells. For training purposes, they split training data such that 100 characters are processed at a time and used dropout = 0.7. Text generated using these parameters showed the most amount of similarity to natural text, for the value of Temperature = 1 [1]. We generated LSTM based text using Keras library, with 2 LSTM layers, each layer containing 250 cells and a dropout = 0.8. For training the LSTMs we used word based approach instead of character based approach, and used 25 words to be processed at a time. Text generated using these parameters can be seen in Table 1 and it shows that as the value of hyper-parameter T increases, the quality of text and its similarity to human generated text increases.

For generating text using OpenAI GPT-2 Y. Qu, P.Liu [2] used a BERT tokenizer and trained the GPT-2 model. The generated model gives accurate readability but generates duplicates [2]. We utilized sample decoding, GPT-2 tokenizer and gpt2-medium, a pre-trained model with 1024 hidden layers, 16 heads and 345 million parameters for OpenAI GPT-2 based text generation. The text generated using these parameters can be seen in Sect. 4.4.2. The text is readable to great extent and unique upto word length = 75, but

some repetitions can be observed for word length = 100. The same text can also be seen to randomly switch topics.

5 Conclusion and Future Scope

In this paper, we generated pseudo text using GPT-2 and LSTMs and evaluated how close this machine-generated text is to human-generated text by checking if they follow statistical features followed by man-made text such as Zipf's and Heap's Laws for Words. We also measured the effect of temperature or randomness on the text generated by LSTM networks. To do this, we tried out various experiments and obtained results, which proved that LSTM and GPT-2 generated texts follow both statistical laws i.e., Zipf's and Heap's law. We also observed that as Temperature (T) increases the quality of text also improves for LSTMs. Although the pseudo text given by LSTMs improves with T, it consists of some grammatical errors and stops making sense after some length, on the other hand, the text generated by GPT-2 is better than the text given by LSTMs in terms of grammar and quality, but lags in terms of computational cost.

This study opens ways for many types of research in the future. The Generative Pretrained Transformer-2 based model performs better than all the models previously used for text generation and generated samples are close to human-generated texts but, it also has some limitations such as because of its huge size it is more computationally expensive compared to previous models, this model gives good performance on generalized topics but performs poorly on scientific or technical data and abrupt changes in the topics can also be noticed. Future researches in the natural language processing community aim to reduce the computational cost of Transformers and LSTMs and to remove the limitations discussed to as much extent as possible.

References

1. Lippi, M., Montemurro, M.A., Degli Esposti, M., Cristadoro, G.: Natural language statistical features of LSTM-generated texts. *IEEE Trans. Neural Netw. Learn. Syst.* **30**(11), 3326–3337 (2019)
2. Qu, Y., Liu, P., Song, W., Liu, L., Cheng, M.: A text generation and prediction system: pretraining on new corpora using BERT and GPT-2. In: 10th International Conference on Electronics Information and Emergency Communication (ICEIEC), Beijing, China, pp. 323–326. IEEE (2020)
3. Santillan, M.C., Azcarraga, A.P.: Poem generation using transformers and Doc2Vec embeddings. In: 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, pp. 1–7. IEEE (2020)
4. Wang, D., Cheng, H., Wang, P., Huang, X., Jian, G.: Zipf's law in passwords *IEEE Trans. Inf. Forensics Secur.* **12**(11), 2776–2791 (2017)
5. Li, C., Su, Y., Liu, W.: Text-to-text generative adversarial networks. In: International Joint Conference on Neural Networks (IJCNN), pp. 1–7. IEEE, Rio de Janeiro (2018)
6. Weizenbaum, J.: ELIZA – a computer program for the study of natural language communication between man and machine. *Commun. ACM* **9**, 36–45 (1966)
7. Gatt, A., Krahmer, E.: Survey of the state of the art in natural language generation: core tasks applications and evaluation. *JAIR* **61** 65–170

8. Godor, B.: World-wide user identification in seven characters with unique number mapping. In: 12th International Telecommunications Network Strategy and Planning Symposium, New Delhi, India, pp. 1–5. IEEE (2006)
9. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N.: Attention is all you need. NIPS (2017). <https://arxiv.org/abs/1706.03762>. Accessed 14 Feb 2021
10. Raghav, B.: Text Generation in NLP - Springboard India. <https://in.springboard.com/blog/text-generation-using-recurrent-neural-networks/>. Accessed 22 Dec 2020
11. Lü, L., Zhang, Z.K., Zhou, T.: Zipf’s law leads to Heaps’ law: analyzing their relation in finite-size systems. PLoS ONE **5** 0014139 (2010)
12. Li, W.: Random texts exhibit Zipf’s-law-like word frequency distribution. IEEE Trans. Inf. Theory **38**(6), 1842–1845 (1992)
13. Otter, D.W., Medina, J.R., Kalita, J.K.: A survey of the usages of deep learning for natural language processing. IEEE Trans. Neural Netw. Learn. Syst. **32**(2), 604–624 (2020)