Shakespeare like Text Generation using LSTM and Tensor flow

[1]Hasan Hussain,[2]Jovin James Maliyakal,[3]Dion Philip,[4]Omprakash Yadav
[1]Student,[2]Student,[3]Student,[4]Professor,
[1]Computer Engineering,
[1]Xavier Institute of Engineering, Mahim, Mumbai, India.

Abstract : This paper aims to helps us understand how to generate Shakespeare like text generation using LSTM and Tensorflow. Our main objective is to learn about Automatic text generation using LSTM and Tensorflow .Automatic text generation is that the text generation of tongue texts by computer. In this project, we are getting to generate words from a given set of input words. We are getting to train the LSTM model using William Shakespeare's writings.

I. INTRODUCTION

Automatic text generation is that the generation of linguistic communication texts by laptop or computer. It's applications in automatic documentation systems, automatic letter writing, automatic report generation, etc. During this project, we tend to square measure attending to generate words given a group of input words. we tend to square measure attending to train the LSTM model victimization William Shakespeare's writings. TensorFlow and Keras are often used for a few wonderful applications of linguistic communication process techniques, as well as the generation of text. During this project, we'll cowl the idea behind text generation employing a perennial Neural Networks, specifically a protracted memory Network, implement this network in Python, and use it to get some text.

II. PROPOSED DESIGN

First, so as to come up with text, we want a dataset. So, for this project, we tend to area unit sonnets that area unit written by William Shakespeare in 1609. Then we'll shut down the information set like deleting the punctuations, changing the capital to minuscule. there's a complete of 896,000 words among the information set except for our project we are going to solely use two hundred,00 words.

Now we tend to area unit attending to import Tokenizer from Keras for knowledge pre-processing. this can be simple: we tend to use the TensorFlow(Keras) Tokenizer category to automatise the tokenization of our drive knowledge. initial we tend to produce the Tokenizer object, providing the most range of words to stay in our vocabulary when tokenization, likewise as associate degree out of vocabulary token to use for encryption check knowledge words we've not stumble upon in our coaching, while not that these previously-unseen words would merely be born from our vocabulary and cryptically unaccounted for. Tokenizer converts the text to whole number format thus it will be used for cubic centimeter. encrypt coaching knowledge sentences into sequences. currently that we've tokenized our knowledge and have a word to numeric illustration mapping of our vocabulary, let's use it to encrypt our sequences. Here, we tend to area unit changing our text sentences from one thing like "My name is Matthew," to one thing like "6 eight a pair of 19", wherever every of these numbers match up within the index to the corresponding words. Since neural networks work by playing computation on numbers, passing in a very bunch of words will not work. Hence, we tend to use sequences. currently ensuing step to create the LSTM and train the model. when the coaching is completed, currently we offer seed text that area unit a primary few words the model can use to come up with the text. Currently the Model will be accustomed Generate any range of words and every time a special set of words area unit generated.

III. METHODOLOGY

- **Tokenize our training knowledge**
  This is straightforward; we tend to exploitation the TensorFlow(Keras) Tokenizer category to automatise the tokenization of our coaching knowledge. 1st we tend to produce the Tokenizer object, providing the utmost variety of words to stay in our vocabulary once tokenization, additionally as Associate in Nursing out of vocabulary token to use for encryption take a look at knowledge words we've got not come upon in our coaching, while not that these previously-unseen words would merely be born from our vocabulary and cryptically unaccounted for. to be told additional regarding different arguments for the TensorFlow tokenizer, examine the documentation. once the Tokenizer has been created, we tend to then work it on the coaching knowledge (we can use it later to suit the testing knowledge as well).
- **Get our training data word index**
  A byproduct of the tokenization method is that the creation of a word index, that maps words in our vocabulary to their numeric illustration, a mapping which can be essential for encryption our sequences. Since we are going to reference this later to print out, we tend to assign it a variable here to alter a touch.
- **Encode training data sentences into sequences**
  Now that we've got tokenized our knowledge and have a word to numeric illustration mapping of our vocabulary, let's use it to encrypt our sequences. Here, we tend to ar changing our text sentences from one thing like "My name is Matthew," to one thing like "6 eight a pair of nineteen," wherever every of these numbers match up within the index to

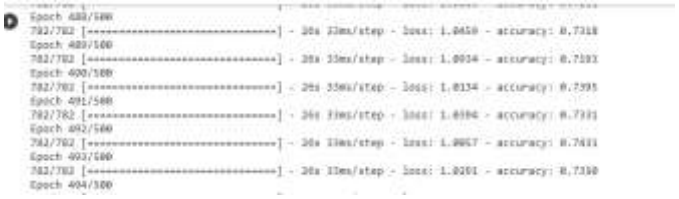
the corresponding words. Since neural networks work by performing arts computation on numbers, passing during a bunch of words will not work. Hence, sequences. And bear in mind that this can be solely the coaching knowledge we tend to at performing on right now; testing knowledge is essentially tokenized and encoded after, below.

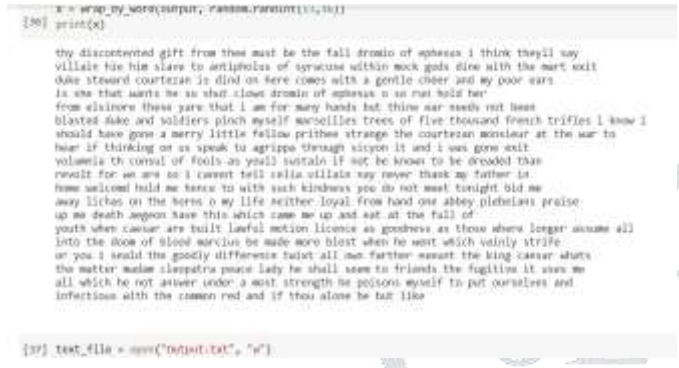
- **Get max training sequence length**
  Remember after we aforesaid we wanted to own a most sequence length for artefact our encoded sentences? we tend to might set this limit ourselves, however in our case we are going to merely realize the longest encoded sequence and use that as our most sequence length. There would definitely be reasons you'd not need to try to to this in observe, however there would even be times it might be applicable. The maxlen variable is then used below within the actual coaching sequence artefact.
- **Pad the training sequences**
  As mentioned higher than, we'd like our encoded sequences to be of a similar length. we tend to simply distinguished the length of the longest sequence, and can use that to pad all different sequences with further '0's at the tip ('post') and can additionally truncate any sequences longer than most length from the tip ('post') additionally.
- **Output the results of our work**
  Now let's have a look at what we've done. we'd expect to notice the longest sequence and also the artefact of these that are shorter. Additionally note that once cushiony, our sequences are born-again from Python lists to Numpy arrays, that is useful since that's what we are going to ultimately feed into our neural network. the form of our coaching sequences matrix is that the variety of sentences (sequences) in our coaching set (4) by the length of our longest sequence (maxlen, or 12).

IV. FLOWCHART

V. RESULTS:

We have gotten final accuracy of 74.6%. The batch size for training was 256 and no of epochs were 500.
We've generated 300 words using our LSTM model. The model are often wont to generated any amount of words 500,1000 or more.
The final generated text are often downloaded within the pc within the sort of document to extend the accuracy we will increase the amount of epochs or we will consider the whole data for training. For this model we've only considered 1/4th of the info for training.





VI. CONCLUSION

So our model will often want to automatically generate words in Shakespeare like English, we will generate any number of words which may be downloaded within the sort of document. You'll be wanting to extend the amount of coaching epochs to enhance the network's performance. However, you'll also want to use either a deeper neural network (add more layers to the network) or a wider network (increase the amount you would like to find out more about how LSTMs work, Learning how the parameters of the model influence the model's performance will assist you choose which parameters or hyper parameters to regulate. you'll also want to read abreast of text processing techniques and tools like those provided by NLTK.

VII. FUTURE SCOPE

We can further improve the accuracy of prediction by increasing the number of epochs or consider the entire data for training. For this project we have considered ¼ of the dataset for training.
The project can be further improved by using a different data set, using NLTK, word2vec, adding extra features like Auto-Capitalization, using a different data set, Full stops after every n words etc.
You could also try adjusting the batch size, one hot-encoding the inputs, padding the input sequences, or combining any number of these ideas.

VIII. REFERENCES

http://www.w3schools.com/
https://stackabuse.com/text-generation-with-python-and-tensorflow-keras/
https://youtu.be/VAMKuRAh2nc
https://kgptalkie.com/text-generation-using-tensorflow-keras-and-lstm/