Text Generation Using Recurrent Neural Networks

Zishen Thajudheen, Amit N Subrahmanya, Aditya Singh, Akshit Jhamb, Vinay Hegde
Computer Science and Engineering, R.V College of Engineering, Bangalore, India

Abstract—**Language Modelling is the core problem for a number of natural language processing tasks such as text generation. In this project, we try to create a language model at a character level for generating natural language text by implement and training state-of-the-art Recurrent Neural Network. No text generation model has been designed for Kannada. With the help of transliteration it is possible to build a text generation model for Kannada. We plan to build this model and analyse its strengths and weaknesses. In this paper we aim to demonstrate the power of large RNNs by applying them to the task of predicting the next character in a stream of text. This is an important problem because a better character-level language model could improve compression of text files. We can evaluate the syntactic and semantic sense of the generated phrases.**

***Keywords*— include at least 5 keywords or phrasesText generation, Transliteration, character prediction, Machine Learning, Recurrent Neural Network (RNN).**

I. INTRODUCTION

With the latest advancements in the field of deep learning many tasks of Natural Language processing are becoming easier and effortless to solve. One such task of Natural Language processing is text generation. Text generation is effectively a language modelling problem. Language modeling is a fundamental problem in Natural Language processing tasks such as text summarization.

Text, a stream of characters lined one after the other. It is particularly difficult to work with generation of text as we need to train the model on the characters and the trained model needs to be very accurate. Even an error of a particular character in the stream of characters could make the whole sentence irrelevant. Text generation using recurrent neural network is working with novels written by authors and sonnets and poems.

The main motivation behind developing this model is to maintain the writing style of the author which cannot be replicated very easily. People are generally writing with emotion and are motivated by some personal experience and it is very hard to replicate the same using a computer. This model does not summaries the text it is trained on. Rather it writes a whole new chapter or poem based on the writing style of the author.

In this paper, we propose a neural network that can successfully generate text. The model can be a character-to-character or word-to-word model. We built, trained and tested character-to-character model and compared the effectiveness and accuracy with word-to-word model. Another barrier we plan to surpass with this model is to have text generation in a language other than English. With the help of encoding characters to numbers and working on the same the model can be trained to write text in a different language like Kannada.

The rest of this correspondence is organized as follows. The paper explains the background of Recurrent Neural Network Model, transliteration and LSTM (Long Sort Term Memory) in section 2. Section 3 presents our proposed method. In section 4, the results of our proposed method for are compared to existing method. All improvements possibly made is mentioned in section 5.

II. BACKGROUND

Recurrent Neural Network is a class of artificial neural network that exhibits dynamic behavior. RNNs can process sequence of inputs by using their internal states and that is why this model is used in text generation, speech recognition, etc. Recurrent Neural Networks differ from feed forward nets because they include a feedback loop, whereby output from previous step is fed back to the network to affect the outcome of the current step and so forth for each subsequent step.

For example, if a net is exposed to a word letter by letter, and it is asked to guess each following letter, the first letter of a word will help determine what a recurrent net thinks the second letter will be, etc. RNN is the neural network model used because the model needs to predict a word based the all the previous words read. That is we need to consider history which is either very difficult or impossible to implement using other neural networks.

Transliteration is another major process used by this model. Transliteration is a process of converting a word from one language to another. To understand better consider a word needs to be transliterated from language x to language y. Unlike translation, transliteration gives an idea of how the word will be pronounced in language x but will be written using alphabets in language y.

RNN with LSTM (Long Short Term Memory) is used in this model. LSTM is used so that it can process multiple data points which is ideal for models working on text and speech. It has a forget gate, input and output gate. LSTM is used to solve the gradient descent problem. Gradient descent problem is a problem that occurs with neural network. That is the gradient tends to get smaller and smaller as we go back in the neurons. It becomes difficult to train based on earlier neurons as they become so small (negligible) compared to the recent weights of the latest neurons. This problem of vanishing gradient can be solved using LSTM.

III. ARCHITECTURE AND IMPLEMENTATION

For text generation we propose a RNN model with Long Sort Term Memory (LSTM). LSTM RNN is used to solve the gradient descent problem. Compared to other neural networks, RNN's have hidden states which help them process past information. This is particularly important considering text generation as we need to process past information to predict new text into consideration and with LSTM the model is capable of long term dependencies. The proposed system has three stages

A. Preprocessing the dataset

Preprocessing the dataset includes various steps. Data cleaning is a process wherein which you remove irrelevant data from the data set such as roman numbers, non-alphabetical characters etc. With these make sure that the whole data is properly punctuated.

Tokenization is the next part of data cleaning where we tokenize the cleaned data. In character-to-character model we split the whole text into characters and tokenization is done. In word-to-word model we split it word by word.

Padding is a process wherein redundant data is added to the tokens so that all the tokens be of the same size. After padding is done all the tokens must be assigned to unique numbers. Considering the model is a neural network model it is always easier to work with numbers. Since we are working with numbers and converting back to relevant characters language is generally not a barrier. Thus building a model for text generation for Kannada characters should not be an issue.

B. Training the data

We deal with training a model where the input are characters of a novel. The amount of input present in the model is very high and as a result a system with a decent GPU will speed up the training process. The data was collected from various sources. Most of them are classical texts that are cleaned by the model. Different models are trained and tested considering accuracy and speed into consideration and the best model is considered.

C. Generating text

Finally the trained model is used for generation of text. A few sequence of characters are given as input and based on these characters and trained model the text are generated. Various models were considered. All of these were trained and then finally the one which had the required efficiency was selected.

For text generation in Kannada we use the process of transliteration. Since we developed a model for text prediction in English we use the same model for text generation in Kannada. A small code was written which acts a dictionary that transliterates English to Kannada. A unique English alphabet was assigned to every Kannada character. Same letters of English characters with different cases (upper and lower case) were assigned to different Kannada characters. Some rules of Kannada language were also taken into consideration when transliterating English to Kannada. The Kannada text is the given as input to this program which transliterates all Kannada characters to English. After this process we then use the same model designed for English to this transliterated text file. The model then trains and the required text is generated. Once the text is generated the model then gives this English text as input to the dictionary program that transliterates text back to Kannada. Once they are converted this text is given as output. The output is fairly accurate and was able to replicate the writing style of the Kannada authors too.

IV. FIGURES

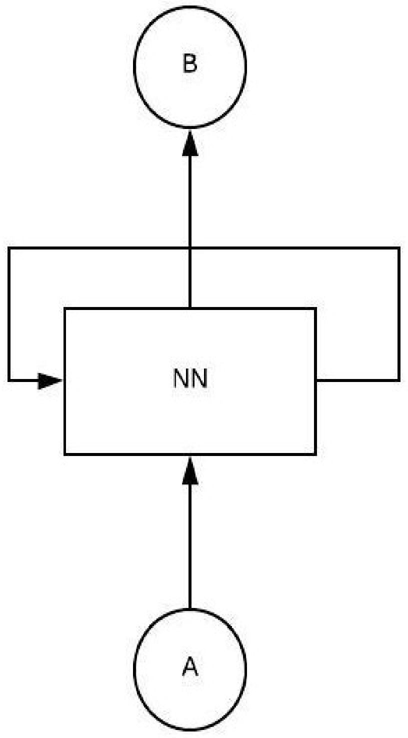

Fig 1: Recurrent neural network structure

Figure 1 represents the basic structure of a recurrent neural network. The neuron is NN and inputs move from A to output B. A recurrent neural network is basically multiple instances of the same network passing something to the successor. So if you open up the neural network structure below it opens up to a chain like structure. This is the basic structure of RNN which had a huge success when working with text, speech and other kinds of data.

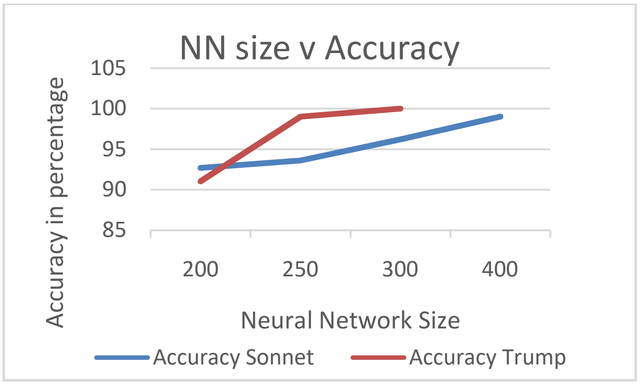

Fig 2: NN size vs Accuracy graph

For an epoch of 200 and depth of 3 the model was varied of the neural netwrok size (NN size) and the accuracy of the model was noted down as shown in Figure 2. Accuracy was calculated by matching the words predicted by the model along with the words available in the dictionary. The dropout value was 0.2 . As it can be noticed that ad NN size increases the accuracy of the model also increases. Accuracy can also reach 100% but the more important thing here is to make sure that the words generated by the model should make logical sense. In the graph above the blue line is the accuracy measured for a shakespeare sonnet text file used. The text file was cleanedby removing the roman numerals from the text file. The second text file used are trump tweets of the year 2018. With both these text files the accuracy finally achieved was awlays greater than 90%.

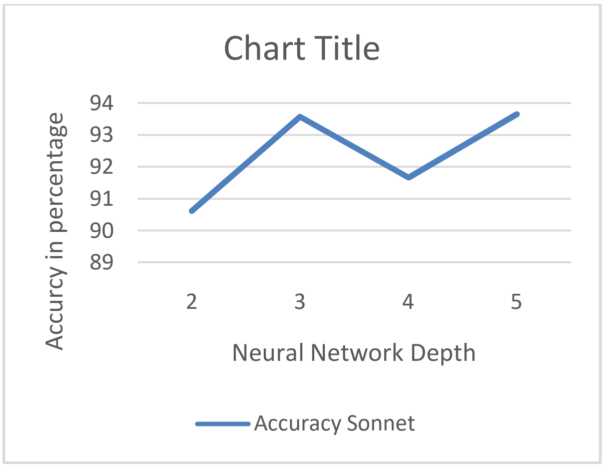

Fig 3: Depth vs accuracy graph

Figure 3 is the depth vs accuracy graph as shown. Here the accuracy does increase as depth increases but not in all instances. So it is clear than it is not necessary that when depth and neural network size increases, accuracy will also increase. Thus we need to keep on training different architectures and then come to the conclusion of the most efficient model both in acuuracy of word correctness as well as how much sense the sentence as a whole would make. For calculation of depth we consider the same files as considered for NN size vs accuracy graph. Epoch of 40 and NN size of 200 were kept constant and we kept on increasing the depth of the model. The figure 2 is the representation of the same.

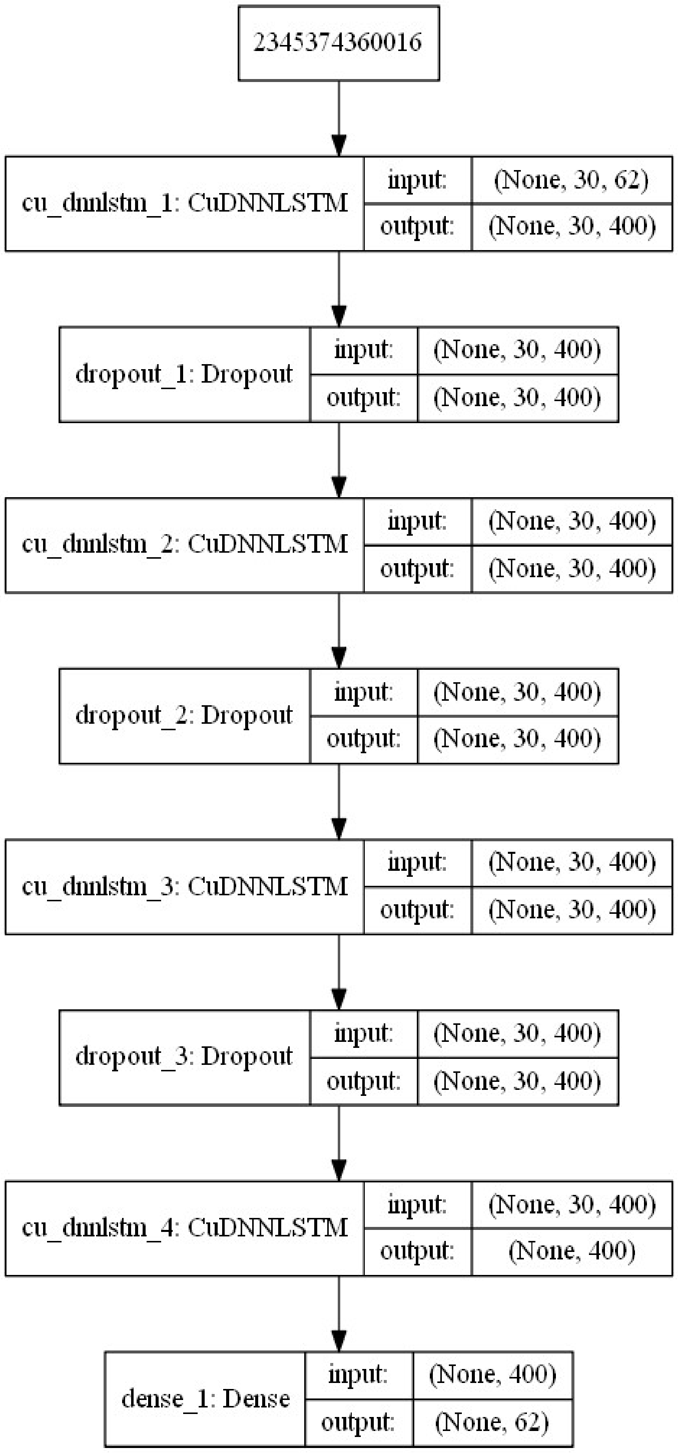

Fig 4: Architecture of the model

Figure 4 is the basic architecure of one of the trained models. Dropout of 0.2 is kept to avoid overtraining. Neural Network depth of 400. Significant amount of time was taken to train this model. Training did use GPU but still did take some time. Resulted in an accurate model which predicted text properly.

V. RESULTS

Various corpus were taken and cleaned by the model. Many classical novels, poems etc. were cleaned and trained on the final model. With all the corpus the text generated by the model is able to capture the writing style of the authors to an extent. The resultant sentences generated by the model had no mistakes and all the words generated did fit together for the sentence to have a meaning.

Input text were both taken in English as well as Kannada. For both these languages the model was able to fare great results. Small changes were to be made to the model which was trained in Kannada datasets to incorporate the language rules present in Kannada which is absent in English.

Comparing the character-to-character model (proposed model) to word-to word model, the training time was comparable between the two if they are trained with the assistance of a GPU.

Comparing the final result the character-to-character model was able to generate better results with the same corpus. More sensible words were generated by character-to-character model.

Since we worked on both character-to-character model and word-to-word model, a faster way for better output was to take the input character and use the character-to-character model to predict a word and then use the word-to-word model to predict the rest of the sentence. On doing so we noticed a fairy accurate result. On top of that it was significantly fast in predicting the sentence too.

In the text generation model build, there is scope for 4 outputs. 4 predictions are made by the model based on the input. The user is free to select whichever output he wants. The accuracy of the prediction decreases as we chose the 4th output as compared to the first output. It is to be noted that all 4 outputs generated by the model are from 4 different models. Thus a variety of outputs will be generated and the end user can choose whichever they feel is relevant to them.

VI. FUTURE WORK

In future work the model could consider all the grammar rules of Kannada that is not present in English. Include all synaptic and semantic rules while transliterating Kannada to English. There are quite a lot of rendering to be done when converting English back to Kannada. All of these rendering were not considered. By including all these rendering the final output will be a bit more accurate than it is right now.

Since we were limited by the GPU available (we used GTX 1050ti) by extensive training of the model with a powerful GPU could increase the accuracy higher (currently we achieved 95%) and thus will provide better results.

Coming up with a new and improved way to calculate the accuracy of the model so that rather than just considering the spelling correctness of the word the accuracy also includes to what level the sentences as a whole which has been generated do make sense. Detailed analysis needs to be done to make sure proper use of grammar has been followed by the model predicted output. Based on this analysis further improvements can be bought into the model to further increase the accuracy and if possible the time taken to predict the sentence generated based on the input.

ACKNOWLEDGEMENT

Our thanks to the Dr. Vinay Hegde for the tremendous guidance and support that he has provided towards this project. Providing us with relevant information and insight which helped us build our model effectively and efficiently. Our sincerest gratitude to him for not only lending a helping hand but also providing the required dataset on which the model works without which this project wouldn't be possible.

REFERENCES

[1] Jia Wei; Quiang Zhou; YiciCai; Poet-based Poetry Generation: Controlling Personal Style with Recurrent Neural Networks; International Conference on Computing, Networking and Communications (ICNC); 2018

[2] Zejian Shi; Minyong Shi; Chunfang Li; The prediction of character based on recurrent neural network language model; IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS); 2017

[3] Kyuyeon Hwang; Wonyong Sung; Character-level language modeling with hierarchical recurrent neural networks; IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP); 2017

[4] Tzu-Hsuan Tseng; Tzu-Hsuan Yang; Chia-Ping Chen; Verifying the long-range dependency of RNN language models; International Conference on Asian Language Processing (IALP); 2016

[5] Martin Sundermeyer; Hermann Ney; Ralf Schlüter; From Feedforward to Recurrent LSTM Neural Networks for Language Modeling; IEEE/ACM Transactions on Audio, Speech, and Language Processing; Volume 23; Issue 3; March 2015

[6] IlyaSutskever; James Martens; E Hinton Geoffrey; Generating Text with Recurrent Neural Networks; Proceedings of the 28th International Conference on Machine Learning (ICML-11); 2015.\

[7] Zachary C. Lipton, John Berkowitz, Charles Elkan; A Critical Review of Recurrent Neural Networks for Sequence Learning; Cornell University; arXiv:1506.00019

[8] Wim De Mulder, Steven Bethard, Marie-Francine Moens; A survey on the application of recurrent neural networks to statistical language modeling; Published in Computer Speech & Language 2015