RabindraNet, Creating Literary Works in the Style of Rabindranath Tagore

Asadullah Al Galib
Department of Computer Science and Engineering, M.Sc in CSE
BRAC University
Dhaka, Bangladesh
asadullah.al.galib@g.bracu.ac.bd

Abstract—Bengali literature has a rich history of hundreds of years with luminary figures such as Rabindranath Tagore and Kazi Nazrul Islam. However, analytical works involving the most recent advancements in NLP have barely scratched the surface utilizing the enormous volume of the collected works from the writers of the language. In order to bring attention to the analytical study involving the works of Bengali writers and spearhead the text generation endeavours in the style of existing literature, we are introducing RabindraNet, a character level RNN model with stacked-LSTM layers trained on the works of Rabindranath Tagore to produce literary works in his style for multiple genres. We created an extensive dataset as well by compiling the digitized works of Rabindranath Tagore from authentic online sources and published as open source dataset on data science platform Kaggle.

Index Terms—stacked-lstm, recurrent neural network, nlp, text generation, literature, rabindranath tagore

I. INTRODUCTION

The field of NLP, specially the domain of text generation has seen massive improvements in recent years with more sophisticated and powerful models built on top of recurrent neural network models such as LSTM as well as more recent transformer based models. The potential applications of AI generated text in domains such as healthcare and hospitality sector is boundless. However, there is a significant gap in utilizing these advancements in AI to better understand human literature and carry out literary analysis in terms of themes, concepts, societal values, norms at the time of writing. This is more true for Bengali literature due to the lack of quality datasets available online. As a result, our treasured literary works remain completely out of usage for advanced machine learning techniques.

In order to bridge this gap between NLP and Bengali literature, we are proposing a character level RNN model based on stacked-LSTM to generate various literary works in the style of Rabindranath Tagore. In the process of creating this model, we also recognised the lack of readily available quality datasets of Tagore that provided granular access to individual literary items such as poems, songs, dramas, essays. To mitigate this issue, we set about creating a compilation of all of Tagore’s works in easily accessible formats which can be used for literary analysis as well as inputs for deep learning models.

II. RELATED WORK

Recurrent neural networks, known as RNN are neural networks with hidden state and a recurrence nature which allows them to work with sequential data such as textual data in literature or time series data [3]. However, with very long sequence of textual data, it gets very difficult to train RNNs due to the vanishing or exploding gradient problem, where it gets difficult to pass on gradient backward for training purpose over longer sequence of training data. To solve this problem, various approaches have been applied to effectively train RNNs over longer sequence of data. One of these approaches is to create gated paths for the gradients to travel over long sequence of neural network units without much change. Long short term memory or LSTM and Gated recurrent unit, also known as GRU use this gate mechanism to be effectively trained on datasets with longer sequence. Recurrent neural networks and its variants have been used extensively in text generation due to their ability to handle sequential data most effectively [3]. More recent advancements in text generation have come from auto-encoders and generative adversarial networks [3]. Character level RNN with Hessian-Free optimizer and comparatively small number of hidden units produced high quality text with sophisticated grammatical structure [4]. RNNs have been used to classify poem meters in order to better understand the literary style of the poetry in arabic literature [5]. Recently Transformer based models such as BERT and GPT-2/3 have shown great promise in generating extremely high quality text [6].

III. DATASET COMPILATION

A. Crawling & Extraction

One of the main hindrances to advanced literary analysis in Bengali literature using machine learning is the lack of quality datasets in user friendly formats which can be used for multitude of tasks such as exploratory data analysis which demands granular access to individual songs, poems or dramas, at the same time aggregated corpus which can be then fed into deep learning models for prediction or classification tasks. At the start of this project we could not find any open source corpus of Tagore that provided both individual and aggregated content of all of his literary works. To fill in this gap of an open source dataset of the aforementioned type, we initiated a
A compilation project to collect and prepare a dataset of Tagore that would contain complete collection of works for all of his genres. We found the digitized collection of Tagore’s work published online by the Department of Information Technology & Electronics, Government of West Bengal, India [2] to be an excellent source to compile the target dataset from. Their online collection included entire works of Tagore in the following genres -

- Novel
- Poem
- Song
- Story
- Essay
- Drama
- Miscellaneous

Each of the genres included collection of individual items. To extract the raw content from the website by maintaining proper hierarchy of items, we created a versatile crawler to handle multiple use cases present in the underlying raw HTML content. The crawler adjusted its extraction methods based on different genres. The entire crawling process includes extraction of collection items, extraction of individual items inside each collection, extraction of paginated content of individual items and storing data in raw textual format after parsing from HTML (see Figure 1).

B. Preprocessing & Formatting

After collecting individual literary items in raw textual format from each genre, we placed them in hierarchy by maintaining proper genre and parent collection. After that, we executed an aggregator that compiled all individual items of a genre into a single file. Before adding each item into corresponding genre, we passed raw text through a basic pre-processor which removed title names, page numbers and other special symbols using regex rules. We prepared two file formats for each genre, CSV and TXT. The CSV format contained comma separated values including literary item name, parent collection, genre and content. This type of format is useful for various exploratory analysis. On the other hand TXT format contained aggregated content from all individual literary items concatenated together which is useful for training deep learning models. We then published this dataset in the data science competition platform Kaggle for the benefit of the community [1].

IV. METHODOLOGY

We used the prepared dataset from previous step [1] to train a character level RNN model consisting of single LSTM layer. For initial step, we trained the model only on poem dataset with epoch set to 40. After training the model, we studied the quality of the generated text for poem genre. See Figure 2 for an excerpt of the generated poem.

From the generated text it is evident that the model learned basic word formation, however it failed to learn how to choose consecutive words so that it would form a coherent content. From this finding, we were inspired to explore stacked-LSTM in order to increase the model capacity for abstract understanding over long sequence of data and give the model ability to learn word association better.

Different sections of the process starting from analyzing the dataset to generating new content are described below in details (see Figure 3). In our work, we experimented with datasets from 3 different genres, namely song, poem and story. All the dataset samples provided in the following sections are taken from song genre.

A. Analyze Dataset

At first we analyzed the dataset to find out basic information regarding the dataset at hand. The song dataset had total 733280 characters. Following is an excerpt form the dataset (see Figure 4).

The dataset had 100 unique characters. Some of which are listed below (see Figure 5).
B. Process Dataset

At this stage, the dataset consisted of sequence of characters. In order to feed the data into our training model we had to convert the dataset from a sequence of characters to a sequence of numbers. To achieve that, we first created a vocabulary or mapping of character-to-index using the unique character extracted in the previous step and replaced all the characters in the dataset with corresponding index numbers. Followings are some samples from the character to index mapping.

```
""" : 3,
'( ' : 4,
') ' : 5,
"' : 6,
'-' : 7,
', ' : 8,
';' : 9,
';' : 10,
'?' : 11,
'[ ' : 12,
']' : 13,
```

We also created an index-to-character mapping to be used during the text generation step to convert the predicted index number to corresponding character.

C. Create Training Batch

To create training batch, we first divided the dataset into sequences of 101 characters. For our experiment, we considered sequence length as 100. The additional character in the sequence was used during the shifting of each sequence to generate input and corresponding output sequence. After this step, the dataset had altogether 7260 sequences of length 101 for each epoch. After that, we created input-output sequences from each sequence in the dataset by removing the trailing index for the input sequence and first index for the output sequence. The result is 7260 input and output sequence pairs of length 100. See Figure 6 for an example.

During training, at the first step, the model receives the index number for first character as the input and uses that to predict the index number for the next character. During next step, model receives the index number of the next character and uses that and previous step context to predict the index for next character (see Figure 7). To run the training process efficiently, we shuffled the dataset and divided into batch size of 64.

D. Build Model

Finally, we built the model using Embedding layer as our first input layer, then we stacked 3 LSTM layers as our RNN layers before adding a Dense layer as the final and output layer. We set the embedding dimension of the first layer to 256 and used 1024 RNN units for each LSTM layer. The output dimension was the vocabulary size which was 100 for the song dataset. Followings are the model summary (see Figure 8) and structure (see Figure 9).

For each character, the model passes it into the embedding layer to get the embedding of stated dimension and executes the LSTM for one step to generate the log likelihood of the next character using the output dimension of vocabulary length. Before training, the model does not have any knowledge of the dataset. See Figure 10 for a sample of generated using initial parameters of the model before the training step.

E. Train Model

We treated the training stage similar to a classification problem. Given the state of the model up to previous time step and current input character, the model had to predict the
class of next character which had the dimension of 100, the vocabulary size. To train the model, we used Adam optimizer and Categorical cross entropy as the loss function and ran it for 40 epochs for all datasets.

F. Generate New Content

After the model had been trained, we generated new text by providing it with a starting sequence of characters and the number of characters to generate. We also used a temperature variable to control the predictability of next character selection. For each generated character, we use it as the previous character and feed it into the model to generate the next one.

V. RESULTS

As previously mentioned, we executed the same training procedure with 3 different datasets from different genres, namely song, poem and story. First we present the output of the model after it was trained on poem dataset for single-layer LSTM (see Figure 11) and 3-layer stacked-LSTM (see Figure 12) with temperature value set to 1 and starting text as "Aji Amar".

We can observe that stacked LSTM model not only learned how to form correct words but the generated text seems more coherent than the single layer LSTM model. However when we provided the stacked-LSTM model with same starting sequence of characters but temperature value of 1.5, it produced more surprising text (see Figure 13).

Next, we present the output of the stacked-LSTM model after it was trained on song dataset with different temperature value (see Figure 14, 15).

We can notice the 3-fold increase in elapsed time during training from single-layer LSTM to stacked-LSTM with epoch=40.

- Elapsed time for training the single later LSTM with epoch=40:
  - Poem: 3113s
  - Story: 3095s

- Elapsed time for training the 3-Layer stacked LSTM with epoch=40:
  - Poem: 9934s
  - Story: 10175s

VI. FUTURE WORKS

In future, we wish to train the stacked-LSTM model for larger epoch values and evaluate the generated text with evaluation metrics other than human perception using literary analysis. We also hope to experiment with other generative models and compare the results with the current LSTM model. Finally, we are working towards publishing this model for general public to explore and enjoy by generating texts...
VII. CONCLUSION

We have taken the first step of diving into Bengali literature and experimenting with text generation in the style of the literary giant Rabindranath Tagore using character level RNN model with stacked-LSTM layers. We have shown that simple stacked LSTM layers can produce high quality text in literary genres such as poem, song and story. This is promising to take on the challenges of much longer texts involving dramas or essays. Gradually we hope to work on more advanced theme analysis and comparative analysis of literary styles of Bengali writers using more sophisticated NLP tools.

REFERENCES

[1] Galib, A. A. (2021). Complete Works of Rabindranath Tagore. Retrieved December 12, 2021, from https://www.kaggle.com/aagalib/complete-works-of-rabindranath-tagore

[2] Digitized works of Rabindranath Tagore, from https://www.kaggle.com/aagalib/complete-works-of-rabindranath-tagore

[3] Iqbala, T., Qureshia S. (2020). The survey: Text generation models in deep learning. DOI: 10.1016/j.jksuci.2020.04.001

[4] Sutskever, I., Martens, J., Hinton, G. (2011). Generating text with recurrent neural networks.

[5] Yousefa, W. A., Ibrahimea, O. M., Madboulya, T. M., Mahmouda, M. A. (2019). Learning meters of Arabic and English poems with Recurrent Neural Networks: a step forward for language understanding and synthesis

[6] Montesinos, D. M. (2020). Modern methods of text generation.