World Journal of Advanced Research and Reviews

eISSN: 2581-9615 CODEN (USA): WJARAI

Cross Ref DOI: 10.30574/wjarr

Journal homepage: https://wjarr.com/

(Research Article)

Bangla text generation system by incorporating attention in sequence-to-sequence model

Nayan Banik [1,*], Chayti Saha [1], Ikbal Ahmed [2] and Kulsum Akter Shapna [3]

[1] *Department of Computer Science and Engineering, Comilla University, Cumilla, Bangladesh.*

[2] *Department of Computer Science and Engineering, CCN University of Science & Technology, Cumilla, Bangladesh.*

[3] *Department of Statistics, Comilla University, Cumilla, Bangladesh.*

World Journal of Advanced Research and Reviews, 2022, 14(01), 080–094

Publication history: Received on 23 January 2022; revised on 30 March 2022; accepted on 01 April 2022

Article DOI: https://doi.org/10.30574/wjarr.2022.14.1.0292

Abstract

In this AI-driven digital era, the pervasive nature of digital data is possible due to the widespread and cheap access to the Internet. Internet is continuously flourishing with data in many forms. Among them, textual data are a great source of information where people share their expressions in written format. Social media, blogs, online newspapers, government documents are some notable textual data sources. Information extraction from this enormous amount of data by manual inspection is time-consuming, cumbersome, and sometimes impossible. Natural Language Processing (NLP) is the computational domain for addressing those limitations by solving human language-related problems. Text summarization, Named entity recognition, Question answering are some of them where a common task for a machine is to generate coherent text. In such scenarios, the input is a sequence of text, and the output is also a sequence, but they differ in length. Sequence-to-Sequence (Seq2Seq) is an algorithmic approach to address that scenario by utilizing layers of recurrent units. However, the simple Seq2Seq model fails to capture the long-term relationship on the input sequence. Research shows that the attention mechanism guides the model to concentrate on specific inputs. Existing literature shows a lack of quality research on this text generation problem in the Bangla language, whereas many languages show excellent results. This work aims to develop such a system by incorporating attention to the Seq2Seq model and justifying its applicability by comparing it with baseline models. The model perplexity shows that the system can generate human-level readable text using a preprocessed dataset.

Keywords: Bangla Text Generation; Sequence-To-Sequence; Natural Language Processing; Text Mining

1. Introduction

Language is the medium through which people can share their feelings, thoughts, understandings of any issues, and this originated from the quest of exchanging information at an early stage of the human era. Due to the continuous evolution of humankind and the geographical distribution of human civilization, numerous languages are created [1]. Bangla is a widely used south Asian language with native speakers from Bangladesh, where it is the national language. Moreover, western parts of India also use Bangla as their regional language. Considering the number of speakers and demographical perspective, Bangla is an essential language in the world [2]. With the ever-increasing widespread utilization of digital media like online newspapers, blogs, social platforms, e-commerce sites, and others, the textual Bangla content on the web is getting large. This high volume of data is generated from human and digital devices and is used for building intelligent systems in many domains like business, finance, automobile, medicine, education, entertainment, and whatnot. Natural Language Processing (NLP) is a branch of engineering that deals explicitly with human language text or speech to collect, process, analyze and build innovative decision-making systems. Some exciting yet typical applications of textual NLP-based tools are text classifier, translator, entity identifier, summarizer, answering

* Corresponding author: Nayan Banik
Department of Computer Science and Engineering, Comilla University, Cumilla, Bangladesh.

question system. Automated text generation providing some source text is a crucial component of many of those NLP systems where it produces coherent human-like text as output [3]. Sequence-to-Sequence, shortly known as Seq2Seq, is an algorithmic model that takes an input sequence, processes it, produces an output sequence, and text generation problems can be modeled using it. Inside the Seq2Seq architecture, Recurrent Neural Network (RNN) unit is used. Depending on the RNN unit used, several versions of the Seq2Seq model can be created. Since the default RNN unit is incapable of capturing the long-term relationship of the source text, an attention mechanism is introduced. This work aims to build a Bangla text generation system incorporating attention into Seq2Seq and provides experimental justification for how it improves the system's performance without having attention.

Bangla ranked the sixth most widely used language globally, with 268 million native speakers around the globe. It is the mother tongue of Bangladesh and also a regional language of India. From conducting businesses to teaching students, circulating official documents for entertainment purposes, people use Bangla in every sphere of their life. The use of cheap Internet facilities paves the way for using Bangla now, even in digital form where email, online news portals, feeds on social networking sites, distribution of legal documents, etc., are composed in the Bangla language. This vast amount of digital content in Bangla is an excellent source of qualitative and quantitative analysis [4].

Extraction of information from textual data can be done using a branch of computer science called Natural Language Processing (NLP). NLP is the computational analysis of text and speech to synthesize meanings from text without any continuous intervention from human observers. Language translation, text classification, tokenization, stemming, lemmatizing, parsing, named entity recognition, text summarization, question answering, fact identification, etc., are examples of NLP-based systems. Developing such systems requires a large corpus of textual data, and the availability of Bangla data is a good fit for analysis.

Text generation is an exciting research domain where an NLP system can generate human-level lucid text about an input document. It is also used as a module for other end-to-end NLP systems like text summarization, question answering, chatbots, etc. Sequence-to-Sequence (Seq2Seq) architecture shows exciting results for text generation tasks in many languages [5]. Nevertheless, for longer documents, the model cannot capture the long-term dependencies. Moreover, not every portion of an input sequence is crucial for the output. So, the regular vanilla Seq2Seq model cannot spot that relationship, and hence attention mechanism comes into place [6]. Many languages employ attention in the Seq2Seq model, with some of them in Bangla. However, extensively reviewing the existing works, it can be justified that further research is required to explore this domain [7].

In a nutshell, the noteworthy usage of Bangla over the Internet and high voluminous textual data should be appropriately utilized by sophisticated tools for building intelligent decision-making systems. Bangla text generation using Seq2Seq with attention will help achieve that goal.

2. Material and methods

2.1. Literature Review

Conducting research requires the exploration of existing works in that particular domain. It helps to understand the ongoing trends and contribute as necessary. Moreover, past research paves the way for future works. Natural Language Processing (NLP) is a long researched domain where many researchers dedicate their knowledge to enhancing the linguistic capabilities of machines. The chapter first reviews the scholarly works in classical approaches in Bangla text generation. Then gradually, it discusses the contribution of researchers in the machine learning-based Bangla language domain for text generation. For each paper, corresponding methodologies, working domain, used dataset, limitations are discussed.

2.1.1. Classical Approaches in Bangla Text Generation

Most of the works in classical text generation aim for extractive summarization. In [8], authors extract information from a textual document by heuristic rules. They performed several standard preprocessing steps like tokenization, stemming, stop words removal. Then the prime sentences were extracted by word and sentence level analyses. For this analysis, they used sentence distance, length, and values. The authors used sentence scoring methods to evaluate the prime sentences, and finally, sentence ranking was employed. For evaluation, they used a comparative score between 0 to 5 for extracted sentences and human-created summaries. Though they claimed to achieve 4.3 for their methods where the human score was 4.6, the system of extractive text generation is limited to the provided limited data. Similar to this, authors in [9] proposed another extraction-based summarization in Bangla. They used collected Bangla online news articles and applied them to preprocess steps similar to the previous paper. The sentence ranking was performed by word frequency (number of the appearance of the word), positional value (where word or sentence was positioned in the document), cue words, and the document structure (header, title, etc.). According to those features, the summary is extracted. They used only 45 news samples and claimed to achieve an F1 score of 83.57%. Moreover, a comparison of the model-generated summaries with human proposed summaries was discussed. However, due to the small number of samples, the work is minimal for real-time analysis.

2.1.2. Machine Learning Approaches in Bangla Text Generation

For machine learning-based approaches, authors in [10] proposed a single document Bangla text summarization that was opinion-oriented. They used a topic-sentiment model with the concept of theme clustering. The dataset was created by collecting news articles from an Indian online newspaper. Several linguistic features were used for the model from the collected data, including lexicographical, syntactic, discourse, etc. They also analyzed POS tagging, WordNet, Frequency analysis, Stemming, Chunking, Dependency parsing, Document structures, and Collocation. Using TF-IDF, authors explored the Conditional Random Field (CRF) to capture the theme of a document. They also applied a clustering algorithm, and the summaries were generated based on IR value. In the paper, the evaluation value was reported as precision 83.60%, F-score 79.85% and Recall 76.44% for theme detection and precision 72.15%, F-score 69.65% and Recall 67.32% for text summarization. Nevertheless, the hierarchical time frame-based analysis is not helpful in a large-scale text generation system. Clustering-based techniques were proposed in [11] where the authors used the K-means algorithm for extractive text summarization. Their sentence clustering approach was claimed to be used for both single and multiple document scenarios. After applying some common preprocessing steps like noise removal, tokenization, and stemming, they applied TF-IDF for word scoring. From that, the sentence scoring was done by the main cue words. The same approach was followed for multiple documents, and based on those values, clustering was done. Among the created clusters, n number of sentences were selected that contributed 30% of the actual input documents. The limiting factor of this work was the non-synchronized nature of generated sentences in summary, and they claimed to improve this with a complex model. In another recent work in [12], the authors proposed a pre-trained language model for unsupervised text summarization in Bangla. They performed standard preprocessing steps by Natural Language Toolkit (NLTK) snd clustered similar sentences. For clustering purposes, a pre-trained model called ULMFiT was used because of the limitations of TF-IDF. Moreover, the silhouette score determined the similarity of the cluster. The sentences that created the summaries were taken from a word graph. For the evaluation, they used Rouge scores and human scores. The limiting factor of the work was that the model was incapable of generating new words. In [13], the authors proposed a Bangla text generation system using LSTM units in the Seq2Seq model. They collected online Bangla newspapers texts with an average sentence count of 4500. Then they removed punctuations, whitespace characters and converted the articles into the utf-8 format before feeding them into the model. The authors did not mention their model structures and evaluation metrics. Moreover, the training was done with news documents for one week. These limiting factors require further investigation. Similar to that, authors in [14] proposed bi-directional RNN for Bangla text generation, claiming to achieve 98.766% accuracy with a pre-trained word embeddings model. The authors did not mention their data sources but mentioned that they had used cleaned data for analysis. Then they converted the data into n-gram sequences and pad those sequences to make them uniform. The authors trained a word2vec model and fed the vectors to LSTM and the Bi-directional LSTM model. Despite the high performance, the lack of dataset statistics and incomplete working method requires more attention to this work. Another work in [15] applied RNN based technique to generate text in an abstractive summarization task. The authors mentioned that they used social media text for the analysis. They cleaned the data and converted them to an n-gram sequence similar to the previously discussed paper. The sequences were then padded, and the LSTM model was trained. The evaluation of the work showed 97% accuracy with 0.0132 loss. However, the work did not experiment with GRU and random length sequence output, limiting their proposed work.

2.2. Methodology

This section aims to provide a theoretical and technical description of the proposed methodology. The systematic elaboration of each module and its justification helps to understand the whole system and hence the section starts with a high-level overview of the system. The overview section outlines the chronological steps performed in this work as a block diagram. Data play a crucial role in any technical research. So, detailed data statistics, including its source, metadata, samples properties, preprocessing steps are described in the next section. The proposed deep learning framework with its constituent parts, including Recurrent Neural Network (RNN) units called Long Short Term Memory (LSTM), Gated Recurrent Unit (GRU), model parameters, training-testing procedures, are described later. The section concludes with the evaluation metrics used in this work.

The high-level system overview is depicted in Figure 1. From the left of the figure, it can be seen that the first data collection step takes place in the system. Almost every real-time dataset contains noises, and hence appropriate data processing steps are applied. In this case, those steps are the removal of non-Bangla, unique and extraneous punctuation characters. Then the text is tokenized, and some data filtering is applied. The processed dataset is then split into train and test data. From the train split, another train and validation split is done where the train split is used to train the deep learning model, and the validation split is used to tune the hyper-parameter of the model. In every time step, the loss is monitored. If it is reducing, more training steps are done, and if not, the final loss is reported bypassing the test data to the model. Finally, the model is deployed on a web server for end-users to use this conveniently.

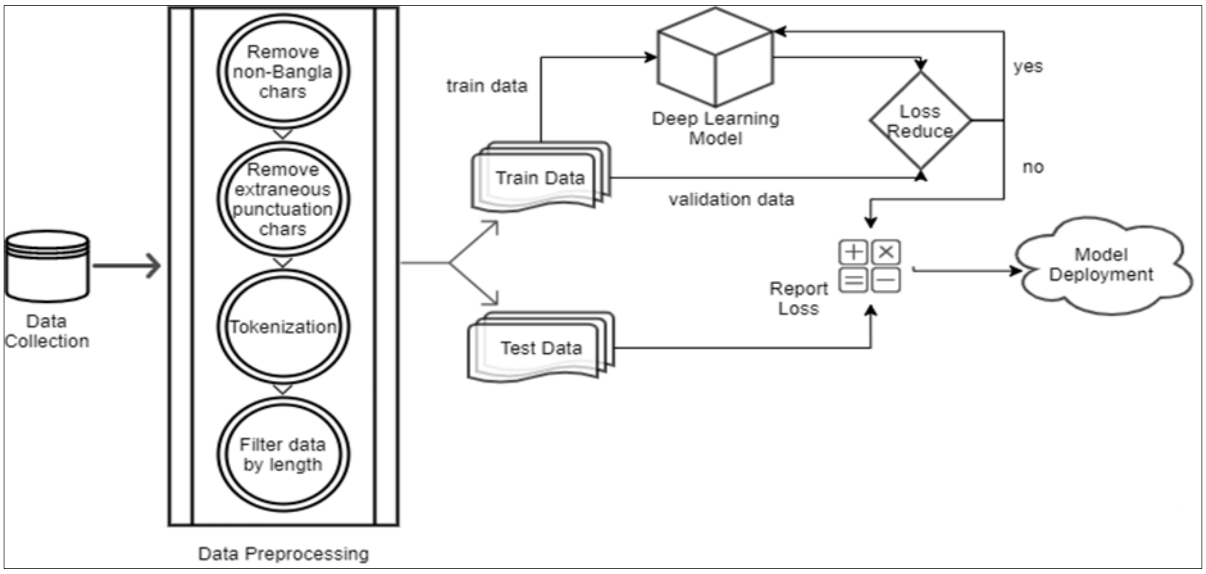

Figure 1 Overview of the Proposed System

2.2.1. Data Collection

The Seq2Seq model or any deep learning-based model requires a large representative dataset to capture the underlying meaning for better generalization. Addressing that, an adequate amount of Bangla text was collected from Bangla Newspaper Dataset hosted on Kaggle [16]. The 6 GB-sized dataset was in JSON format created by crawling the Prothom-alo archive, a popular newspaper in Bangladesh from the year 2013 to 2019 containing 4,37,948 samples. In addition to the news title and content, each sample contained several metadata of thorough news coverage, e.g., author, category information, number of comments on that news, publication and modification date, tags, and URL. From them, except comment count, all other attributes were in string format. The metadata list was essential for more granular analysis, but only content and title worked as features in the proposed work; hence, those two columns were preprocessed accordingly.

2.2.2. Data Preparation

Typically, news articles are written in a standard language form following all the rules of grammar and semantics. However, since the dataset is created by crawling the news website, unwanted noise might be included and requires to be removed. First, the non-Bangla characters are removed from the data. It includes English characters, HTML tags, different control characters, special characters like smileys and emoticons, etc. Then the non-essential and excessive punctuation characters are removed except for the sentence terminator od 'dari' in Bangla text. Both news content and headlines are tokenized, and their length filters out the samples.

Removal of non-Bangla characters

The news data often contained non-Bangla characters, including English character sets (A-Z, a-z, 0-9), control characters (carriage return, line feed, newline), graphical symbols (smileys, emoticons), punctuations (period, commas, hyphens, colons, quotes), Unicode sequences (\u0048, \xa55), excessive spaces. Such characters did not convey any generalizing power to the model and hence required to be removed. Moreover, the removal of those kept the character set clean as well as the feature space to be dense.

Removal of punctuation characters

In news articles, chances of special punctuation characters like hyphens. Colons, underlines, etc., are few unless the news contains dialogs in it. Nevertheless, most of the news contains the author's perspectives; hence, these punctuations are not crucial for analysis. Moreover, while the removal of non-Bangla characters occurs, there might be a situation where words are separated by more than one whitespace. These excessive characters should be removed too. It makes the dataset more concise and less sensitive to noises.

Tokenization of words

Tokenization is the process of dividing the sentences into words which are the primary features of the model. Since the Bangla language has a simple sentence structure with each word separated by whitespaces, the python string split function is used in this work. Moreover, the sentence terminator character ('dari') is associated with the very last word. So, it is first separated from the word by including whitespace between the word and the terminator and then tokenized.

Filter data by length

Often the dataset for training the machine learning model contains excessive samples having correlations and outliers. It is also required to make the data as uniform as possible so that the variation of data conveys variation in meaning; rather than the content length. For this exploratory analysis, the original dataset statistics after applying the aforementioned preprocessing showed that both content length and headline length has a large range. So, considering the median and standard deviation, content length ranging from 200 to 350 words and headline ranging from 4 to 8 words is used in this work. The filtered dataset statistics is given in Table 1.

Table 1 Filtered Data Statistic

Properties	**Content Length**	**Headline Length**
Number of samples	123470.00	123470.00
Average number of words in sample	220.54	5.65
Standard deviation of words in sample	43.28	1.21
Minimum number of words in sample	150.00	4.00
25th percentile value	183.00	5.00
50th percentile value (median)	218.00	6.00
75th percentile value	257.00	7.00
Maximum number of words in sample	300.00	8.00

2.2.3. Seq2Seq overview

Sequence-to-Sequence models, widely used as Seq2Seq models, consist of two separate but interrelated modules known as encoder and decoder. Since every Seq2Seq has this architecture in common, they are also called encoder-decoder models [17]. Both encoder and decoder use a Recurrent Neural Network (RNN) where the encoder encodes the given source sequence (input) into a single vector and the decoder decodes that into the target sequence (output). The single vector, also known as the context vector abstracts the meaning of the input sequence provided to the encoder and the decoder utilizes this to generate the target sequence sequentially (one token at a time). A high-level pictorial representation of the Seq2Seq model is depicted in Figure 2.

Here, the input or source Bangla sentence is converted to an embedded vector by passing through the embedding layer (yellow). To denote the beginning and end of the sentence, unique token <sos> (start of a sentence) and <eos> (end of the sentence) are appended to the input sentence. The embedding e of the word x at time step t represented as $e(x_t)$ is sent to the encoder RNN along with the previous hidden state at a t-1 time step, h_{t-1} where it outputs a new hidden state h_t. So, the RNN can be modeled as a function:

$$h_t = \text{EncRNN}(e(x_t), h_{t-1})$$

Considering the input sentence $X = \{x_1, x_2, x_3,,\dots, x_T\}$ having x_1=<sos>, x_T=<eos> and so on, the very first hidden state h_0 can be either set to zero, very small random numbers or predefined parameters. The final hidden state h_T upon passing the final word x_T to the encoder works as the context vector z that captures the overall context of the given source sentence.

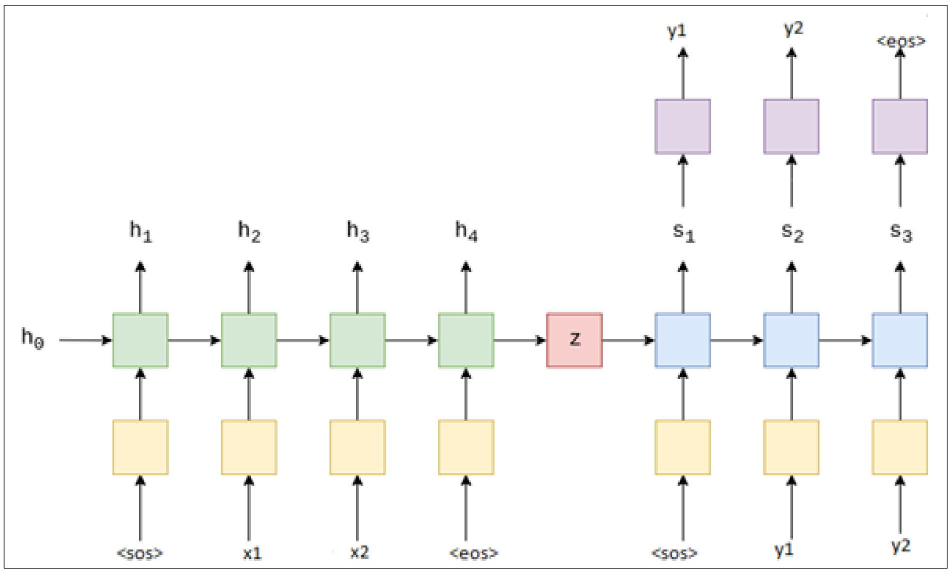

Figure 2 Overview of Seq2Seq model

For decoding the target sentence from context vector z, the decoder network also appends the target sentence marker <sos>, <eos> and passes them to the embedding layer (blue). Similar to the notation used in the encoder, $d(y_t)$ represents the embedding d of word y at time step t. After the embeddings of the target sentence, the embedded vectors are passed to the decoder where the previous hidden state or initial hidden state of decoder s_0 is set to the final hidden state of encoder h_T or the context vector z. In functional form, we can represent the decoder as:

$$s_t = \text{DecRNN}(d(y_t), s_{t-1})$$

Even though both source embedding e and target embedding layers d are in yellow, they are not the same layer. They are two different layers with their parameters.

From the decoder state vector at every time t, state vector to actual word conversion is done by a linear layer (purple). It outputs the next probable word $\hat{y}_t$ for the target sentence $\hat{y}_t = f(s_t)$. The decoder generates one word at a time. The generation begins with the <sos> token passed as the first input decoder y_1. For the later, $y_{t>1}$, outputs from the previous step y_{t-1} or the actual outputs are provided.

In both training and testing, target sentence length is known so that generation can be stopped once the maximum length is achieved or the model generated the end token <eos>.

Once we have our predicted target sentence, $\hat{y}_t = \{\hat{y}_1, \hat{y}_2, ..., \hat{y}_T\}$, we compare it against our actual target sentence, $Y = \{y_1, y_2, ..., y_T\}$, to calculate our loss. Then this loss is used to update all of the parameters in model.

Every Seq2Seq model workflow can be divided into three parts. First, the encoder portion is set up. Then the corresponding decoder is configured. Moreover, finally, the Seq2Seq module combines the two and trains the model with tunable parameters. For the rigorous analysis, several baseline models are described before introducing the attention mechanism.

Layered LSTM-based Model

In this model, layers of LSTM units are used as encoders and decoders in the Seq2Seq model. These multilayer LSTMs can be extended up to any level, but for brevity, a two-layer model is proposed. In a multilayer network, input goes from one layer to another while transforming them in a fashion that the output of one layer becomes the input. For example, after embedding the input sequence X, the embedded vector goes into the first LSTM layer where the output $H = \{h_1, h_2, h_3, ..., h_T\}$ is used as the input to the next layer. To denote this layer-wise concept, the superscript notation can be used in the RNN equation as:

$$h_1^1 = \text{EncRNN}^1(e(x_t), h_{t-1}^1)$$

$$h_t^2 = \text{EncRNN}^2(h_t^1, h_{t-1}^2)$$

This superscript notation can also apply to the layer-wise initial hidden state h_0^l and z^1. Though the high-level representation of RNN only takes data and the previous hidden state as inputs and produces an output, the LSTM unit takes data, previously hidden state, previous cell state as input, and generates hidden state and cell state as outputs. However, that cell state is not going to be the input for the next layer. Denoting cell state as c, previous equations can be written as:

$$(h_t^1, c_t^1) = \text{EncLSTM}^1(e(x_t), (h_{t-1}^1, c_{t-1}^1))$$

$$(h_t^2, c_t^2) = \text{EncLSTM}^2(h_t^1, (h_{t-1}^2, c_{t-1}^2))$$

The encoder takes in some hyperparameters in its formation. The one-hot encoding of unique words (vocabulary) in source sequences works as the input to the encoder. So, the input dimension is the size of a one-hot encoded vector. Each one-hot encoded vector of the input is transformed into an embedding vector, and hence embedding dimension is another critical parameter of the embedding layer. Moreover, the number of layers in the encoder and the number of hidden/cell units per layer are also two tunable parameters. To reduce overfitting, dropout should be applied between layers, and hence it is used as a regularization parameter of the encoder.

The initial hidden state and cell state are set to zero or some small random numbers. As outputs, LSTM returns the top layer time-wise hidden states, stacked layer-wise final hidden state h_T and stacked layer-wise final cell state c_T. However, for constructing the context vector, only final hidden and cell states are required. So, the layer-wise hidden and cell states are ignored.

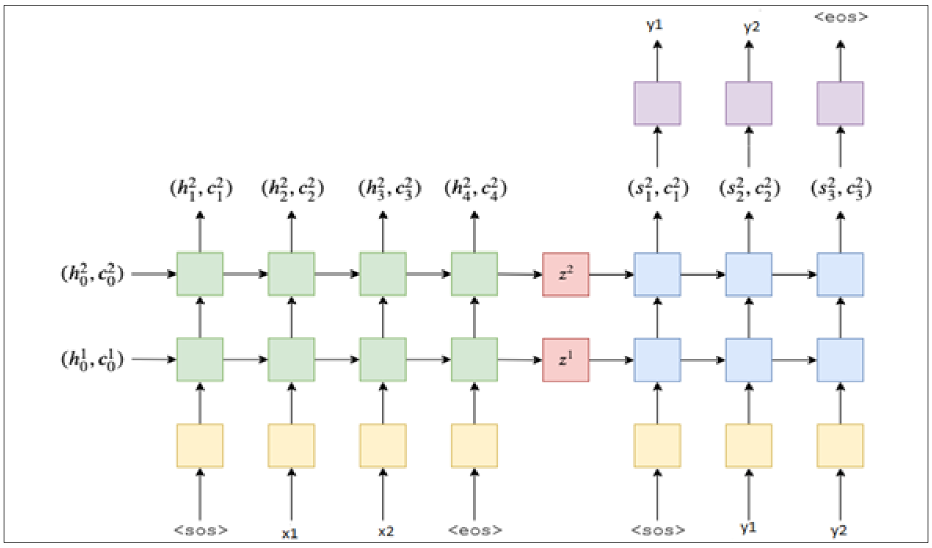

Figure 3 Layered LSTM-based Model Architecture

The decoder decodes the input context vector one token at a time. Analogous to the workings of the encoder, here, the previous hidden and cell states are input to the LSTM unit along with the embedded target sequence. However, the initial hidden and cell states are set as the encoder context vectors, which are the final hidden and cell states returned by the encoder. Considering the hidden state of the decoder as s, this whole scenario can be represented as:

$$(s_t^1, c_t^1) = \text{DecLSTM}^1(d(y_t), (s_{t-1}^1, c_{t-1}^1))$$

$$(s_t^2, c_t^2) = \text{DecLSTM}^2(s_t^1, (s_{t-1}^2, c_{t-1}^2))$$

$$(s_0^1, c_0^1) = z^1 = (h_T^1, c_T^1)$$

Unlike the encoder, the topmost layer output of the decoder is passed through a linear layer f where the subsequent token from the target sequence is predicted.

$$\hat{y}_{t+1} = f(s_t^L)$$

The final structure is shown in Figure 3.

Single Layer GRU-based Model

For encoder and decoder, Gated Recurrent Unit (GRU) [18] is used due to their inherent gated mechanism for capturing meanings from longer sequences and information compression techniques. The encoder part of this model is similar to the previous one with a single change. There is only a single layer of GRU for both encoder and decoder. However, GRU in decoder works differently. Here, the decoder unit utilizes the hidden vector z at each time step to generate the next hidden state.

$$s_t = \text{DecoderGRU}(d(y_t), s_{t-1}, z)$$

As a result, the final linear layer requires the context vector for prediction too.

$$\hat{y}_{t+1} = f(d(y_t), s_t, z)$$

So, the resulting architecture is depicted in Figure 4.

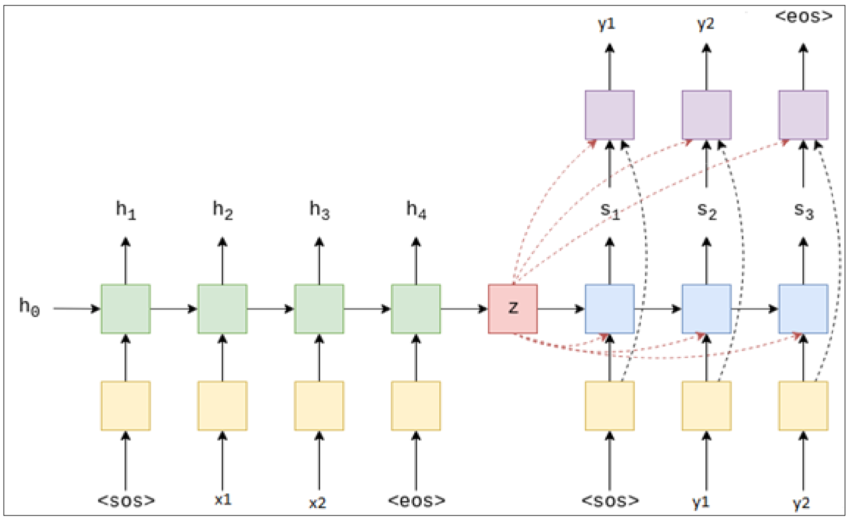

Figure 4 Single Layer GRU-based Model Architecture

Attention-based Model

The encoder part of the attention-based model remains the same as the single layers GRU model. However, the units are bidirectional RNN so that layer can read the input from left to right and another one from right to left. Then the attention layer is defined. This layer takes input as the hidden state of the decoder in the previous time step and all the forward-backward encoder hidden states. It outputs a_t, a vector length equal to the input sentence called the attention vector, where the sum of all values is equal to 1. Moreover, the energy between decoder state and encoder hidden states is calculated by first making a copy of t decoder states for matching the encoder's dimension. It is then passed to the attention layer with tanh activation.

$$E_t = \tanh(\text{attn}(s_{t-1}, H))$$

To match the dimension of the attention vector, it is multiplied by the decoder hidden dimension: $\hat{a}_t = vE_t$. Finally, the attention layer is passed to a softmax layer: $a_t = \text{softmax}(\hat{a}_t)$. The decoder, using the attention layer sums up all the encoder hidden outputs and generate a weighted source vector, w_t as shown in Figure 5: $w_t = a_tH$. Finally, this weight vector is passed to the decoder units and output the states.

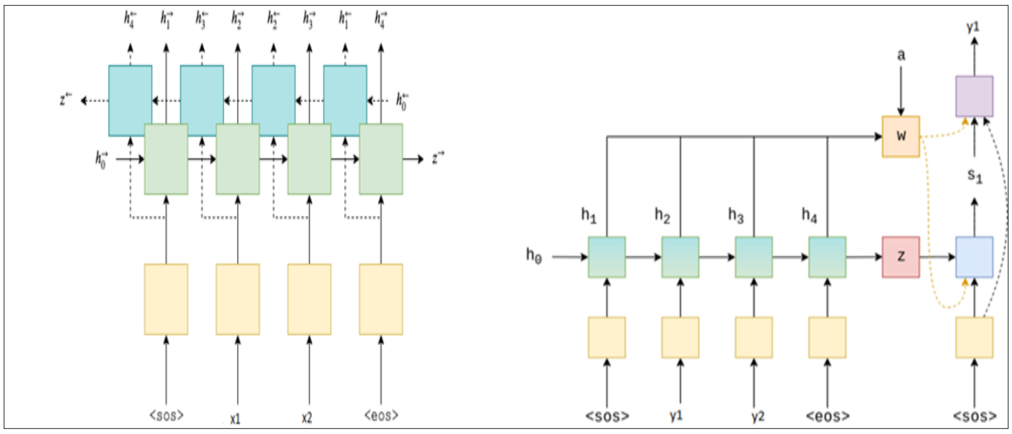

Figure 5 Encoder and Decoder of Attention-based Model Architecture

$$s_t = \text{DecoderGRU}(d(y_t), w_t, s_{t-1})$$

$$\hat{y}_{t+1} = f(d(y_t), w_t, s_t)$$

2.2.4. Evaluation Metrics

For evaluating the performance of the training process and the final model, perplexity is used. Perplexity defines the power of capturing the uncertainty of a language model through a probability distribution. This metric is closely related to the concept of Entropy. If a model is less uncertain or has less Entropy [19], the model will also have less perplexity. So, a well-trained model will provide lower perplexity. For a sentence s with n words, the perplexity will be calculated as:

$$\text{Perplexity}(s) = \sqrt[n]{\frac{1}{p(w_1^n)}}$$

3. Results and discussion

This section provides the experimental results of the proposed work as per the methodology defined in the previous section and analyzes them from several perspectives. The first sub-section reviews the experiment-specific data statistics, including sample counts, sample lengths, words count, and sample proportionality. The following sub-section discusses the required parameters and hyperparameters of the proposed model, including the number of layers, neurons per layer, activation functions, loss metrics, and optimizer properties. Following that, the final sub-section provides the train, validation, test performance of the work in the loss-perplexity curve per epoch and analyzes their behavior in generating texts.

3.1. Experimental Dataset Statistics

As mentioned in the previous section, the collected dataset goes through the preprocessing steps and is cleaned accordingly. The filtered dataset contains 1, 23,470 samples with mean content and headline length of around 220 and 6 words per sample. These tokenized samples are first used to build the vocabulary of the model. Vocabulary indicates the unique words in the content and the headline separately. The total word count also defines the complexity of the model. To train a model and tune its parameters, the whole dataset is split into three parts. The training dataset contains the majority of the samples. It is used to feed the model for learning the inherent meanings or input-output mapping. After a batch of training samples is fed, a validation dataset is used to check for the prediction of that trained model. The model is then optimized by changing the parameters mentioned in the next section to improve the performance. After some epochs, the model performances are saturated, and the final model is applied to the test dataset to report its evaluation. Due to the large number of samples in this work, 90% of data (1, 11,123 samples) is used for training, 5% (6,174 samples) for validation, and 5% (6,173 samples) for testing purposes. Unique tokens reported in training content and headline are 1, 90,181 and 28,062, respectively.

3.2. Model Training

The model is trained on the training split of the whole dataset. Like any deep learning model, several parameters and hyperparameters need to be set up, and their initial values must be configured. For the proposed model, those configuration values are summarized in Table 2.

For training all three models, 128 samples are taken at a time step. This batch size is a hyperparameter of the model, which can be tweaked to fight overfitting in the cost of training time. Each word in news content acts as input to the Seq2Seq model. So, the input dimension of the model is equal to the number of unique words or vocabulary size of the content. Similarly, the model has to predict the words present in the headlines. The output dimension is hence the vocabulary size of the headline. As mentioned in the previous chapter, each word has to be encoded in a vector of predefined size. So, here both for encoder and decoder, the embedding dimension is kept at 100. In other words, the model treats each word as a 100-dimensional vector. The number of neurons on every model in both encoder and decoder is kept the same. Increasing the number of hidden units may increase the complexity of the model and also degrade the performance. So, a moderate value of 256 is set. While training a deep learning model like this Seq2Seq model, specific units may need to be inactive for regularization. It helps the model avoid overfitting and reduce the number of parameters. The model output one word at a time, and it has to predict that word out of all words in the output vocabulary. So, this is a multi-class classification problem, and hence categorical cross-entropy loss is used as the loss function in all the models. This function calculates the difference between probability distribution between two words. For optimization, an Adam optimizer is used. It is a modified stochastic gradient descent algorithm to fight sparse gradients, and this model is a good fit for such a scenario. The learning rate of the optimizer is set to 0.0001. For the first model, all the weights are initialized from -0.08 to 0.08. For the rest of the two models, all the weights are initialized to a normal distribution of mean 0 and a standard deviation of 0.01. The total number of trainable parameters in three models are 30,822,090, 39,772,802 and 47,756,674 respectively.

Table 2 Training parameters/hyper parameters values statistics

Values	Counts
Batch size	128
Input dimension	190181
Output dimension	28062
Encoder embedding dimension	100
Decoder embedding dimension	100
Encoder hidden dimension	256
Decoder hidden dimension	256
Encoder dropout (%)	0.5
Decoder dropout (%)	0.5
Learning rate	0.0001
Epochs	20

3.3. Evaluation of Results

In all three models, 20 epochs are used for training with an early stopping condition. It terminated the training when the validation loss started increasing for two epochs. Since three separate models are defined to compare the performance, they are analyzed separately in the following subsections.

3.3.1. Layered LSTM Model

The training and validation statistics of Layered LSTM model's both loss and perplexity measures per epoch is given in Table 3. The corresponding loss and perplexity trend is visualized in Figure 6.

Table 3 Training/Validation Loss and Perplexity Statistics of Layered LSTM Seq2Seq Model

Epoch	Training Loss	Validation Loss	Training Perplexity	Validation Perplexity
1	7.396	7.102	1630.241	1214.162
2	7.111	7.097	1225.964	1208.849
3	6.869	6.762	962.026	864.61
4	6.52	6.574	678.482	716.497
5	6.208	6.363	496.546	580.082
6	5.938	6.197	379.079	491.257
7	5.686	6.084	294.830	438.726
8	5.474	6.019	238.403	411.241
9	5.3	5.963	200.247	388.728
10	5.135	5.952	169.924	384.491
11	4.988	5.952	146.599	384.395
12	4.853	5.925	128.150	374.287
13	4.732	5.928	113.547	375.319
14	4.613	5.963	100.796	388.624
15	4.501	5.969	90.083	391.21

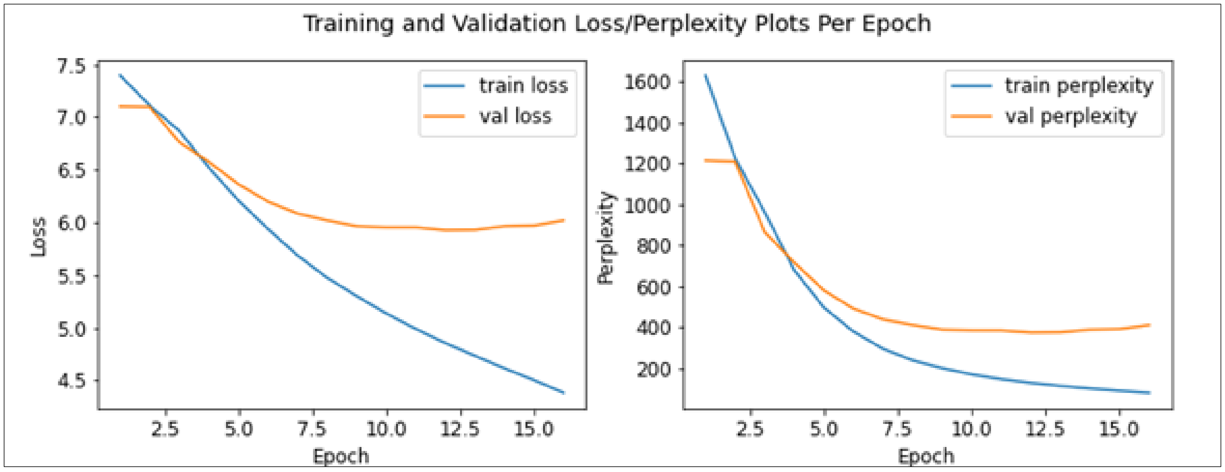

Figure 6 Loss and Perplexity trends of Training/Validation data of Layered-LSTM model

From Table 3, it can be seen that the model started converging from the first epoch for both the training and validation dataset, and the lowest validation loss of 5.925 was achieved at the 12th epoch when the perplexity was recorded at 374.287. Though training loss and perplexity were still decreasing after this epoch, the validation counterparts were increasing, and the best model is recorded for the 12th epoch. In this model, the average time for epoch is recorded as around 6 minutes. From the trend visualization in Figure 6, it is evident that if the model is trained after the validation loss reached the minimum value, the model will start overfitting the training dataset. The final model outputs a test loss of 5.949 and test perplexity of 383.347.

3.3.2. Single Layer GRU Model

The training and validation statistics of Single Layer GRU model's both loss and perplexity measures per epoch is given in Table 4. The corresponding loss and perplexity trend is visualized in Figure 7.

Table 4 Training/Validation Loss and Perplexity Statistics of GRU-Based Seq2Seq Model

Epoch	Training Loss	Validation Loss	Training Perplexity	Validation Perplexity
1	7.384	7.109	1609.612	1223.513
2	6.918	6.775	1009.900	876.112
3	6.418	6.429	612.571	619.716
4	5.908	6.107	367.959	448.944
5	5.464	5.904	236.014	366.399
6	5.101	5.775	164.254	322.298
7	4.754	5.769	116.056	320.062
8	4.461	5.747	86.582	313.233
9	4.197	5.756	66.510	315.924
10	3.941	5.828	51.482	339.584

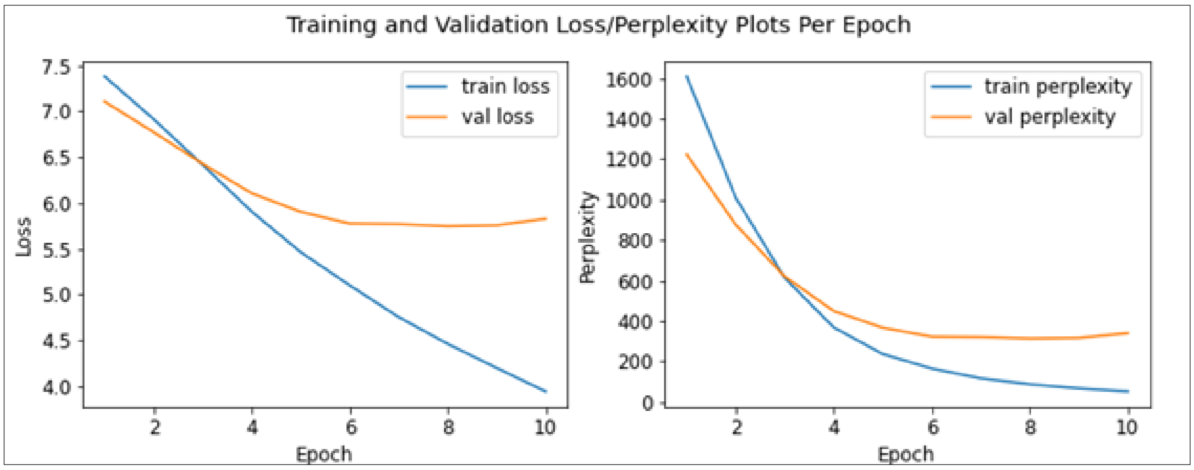

Figure 7 Loss and Perplexity trends of Training/Validation data of GRU-Based Seq2Seq Model

From Table 4, it can be seen that the model started converging from the first epoch for both the training and validation dataset, and the lowest validation loss of 5.747 was achieved at the eighth epoch when the perplexity was recorded at 313.233. Though training loss and perplexity were still decreasing after this epoch, the validation counterparts were increasing, and the best model is recorded for the 8th epoch. In this model, the average time for epoch is recorded as around 5 minutes. So, it can be said that the model trains faster than the previous one and performs better than that. From the trend visualization in Figure 7, it is also evident that if the model is trained after the validation loss reached the minimum value, the model will start overfitting the training dataset. The final model outputs a test loss of 5.743 and test perplexity of 311.959.

3.3.3. Attention Based Model

The training and validation statistics of Attention based model's both loss and perplexity measures per epoch is given in Table 5. The corresponding loss and perplexity trend is visualized in Figure 8.

From Table 5, it can be seen that similar to the previous two models, the model started converging from the first epoch for both the training and validation dataset, and the lowest validation loss of 5.464 was achieved at the 6th epoch when the perplexity was recorded 236.126. Though training loss and perplexity were still decreasing after this epoch, the validation counterparts were increasing, and the best model is recorded for the 6th epoch. In this model, the average time for epoch is recorded as around 12 minutes. So, it can be said that the model trains slower than the others but performs better than that. From the trend visualization in Figure 8, it is also evident that if the model is trained after the validation loss reached the minimum value, the model will start overfitting the training dataset. The final model outputs a test loss of 5.479 and test perplexity of 239.700.

Table 5 Training/Validation Loss and Perplexity Statistics of Attention-Based Seq2Seq Model

Epoch	Training Loss	Validation Loss	Training Perplexity	Validation Perplexity
1	7.22	6.589	1366.879	727.237
2	6.274	6.035	530.638	417.85
3	5.592	5.718	268.164	304.217
4	5.028	5.529	152.643	251.922
5	4.544	5.479	94.059	239.57
6	4.126	5.464	61.944	236.126
7	3.743	5.531	42.212	252.278
8	3.4	5.59	29.966	267.656

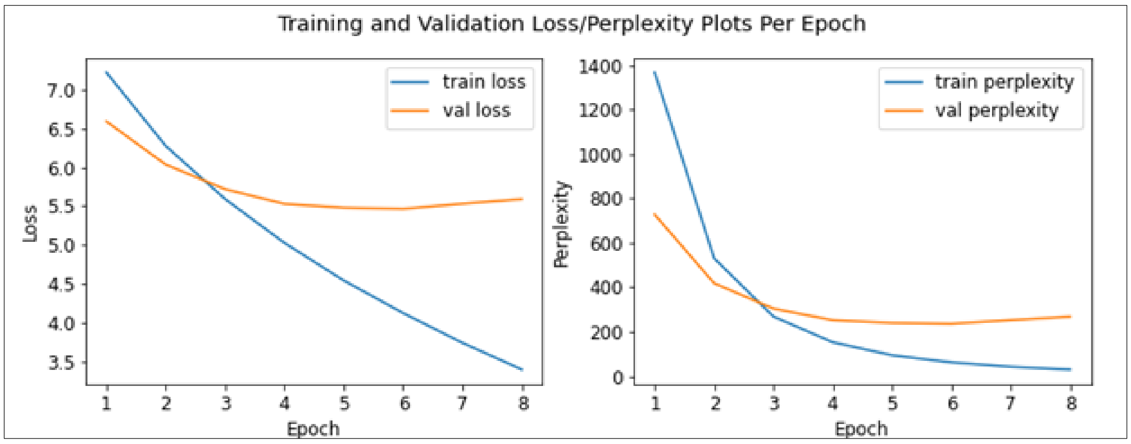

Figure 8 Loss and Perplexity trends of Training/Validation data of Attention-Based Seq2Seq Model

The comparative evaluation of the attention-based model shows that it performs better than the layers LSTM based model and single-layer GRU-based model. For both the non-attention-based models, the perplexity value denotes that those models faced problems to predict the uncertainty as the input sequence is much longer than the output sequence. Moreover, the unknown words in the validation and testing dataset make it more confusing. On the other hand, the attention-based model can concentrate on particular parts of the input sequence and provide better generalization in validation and testing datasets. It justifies the generalization power of this model, and its applicability is the Bangla text generation task.

4. Conclusion

Understanding human language and extracting information from it using the machine is an active research domain for decades. Augmenting that knowledge base, a Bangla text generation system is developed in this work using modern deep learning-based approaches called Sequence-to-Sequence (Seq2Seq) model. Considering the number of active speakers and the usefulness of the Bangla language in many sectors of society, this system will increase productivity for downstream Natural Language Processing (NLP) tasks like text summarization, question answering, title generation, and many more. Bangla language is used by more than 268 million native and non-native speakers globally, and it is the mother tongue of Bangladesh. In day-to-day life, Bangla is necessary for expressing feelings, sharing opinions, and conveying information. Bangla language users are increasing over the Internet, and their shared texts are a great source of information. Extraction of that Information by machines uses different algorithms like Seq2Seq. However, long documents often confuse the Seq2Seq model for generating text. So, an attention mechanism is used. Research in Bangla

NLP is still in a formative stage. Previously, statistical and classical algorithms with heuristic approaches like hand-crafted manual rules were used for different tasks. With technological improvement, many deep learning algorithms are showing promising results in the next generation domain. Exploration of the literature shows minimal work has been done for Bangla, and the performances need to be improved. Bangla news datasets are available on the Internet, but they contain many noises that required to be cleaned. Removal of those noises using standard preprocessing tasks is common in NLP. Similar operations are also performed in this work with the collected Kaggle dataset. A Seq2Seq architecture designs the proposed model with LSTM layers, GRU, and attention layers. Experimental results show that the use of single-layer GRU improves the performance of the layered LSTM architecture. The perplexity value is evident that the attention layer is also better than the bare usage of GRU. The minimum perplexity on the test dataset is reported as 239.70, whereas the previous two models result in perplexity values of 311.95 and 383.34, respectively. The loss/perplexity trends also show that the attention model better converges to the provided dataset in the cost of training time. Overcoming the limitations of current research shows the path to conducting more research in any domain. This holds for this work as well. The use of more data will improve the performance. So, further work can be done if representative data can be availed. Language-specific tools often improve the model like contraction handling, post-processing of stop words, negation mapping, etc. Shifting the trained model to another domain may also become an exciting field to discover. Incorporating a pre-trained word embeddings model for this work can also be an extension for future references. Inspecting the effects of changing hyperparameters is also a reference in further contribution to this domain. The evaluation is done in perplexity, but the modern system does that in Rouge score, BLEU score. So, an investigation can be done to check their values in work. Recent advancements in language-agnostic models like BERT, Roberta, etc., may be explored for this domain as well.

Compliance with ethical standards

Acknowledgments

This research work is funded by Comilla University Research Grant with memorandum no. *CoU/Reg./Research_Project-485/2013/12118 (37)* for the fiscal year 2020-21.

Disclosure of conflict of interest

The authors declare that there is no conflict of interest regarding the publication of this document.

References

[1] Manning C. Understanding human language: Can NLP and deep learning help?. InProceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval 2016 Jul 7 (pp. 1-1).

[2] Sen O, Fuad M, Islam MD, Rabbi J, Hasan MD, Baz M, Masud M, Awal M, Fime AA, Fuad M, Hasan T. Bangla Natural Language Processing: A Comprehensive Review of Classical, Machine Learning, and Deep Learning Based Methods. arXiv preprint arXiv:2105.14875. 2021 May 31.

[3] Alam F, Hasan A, Alam T, Khan A, Tajrin J, Khan N, Chowdhury SA. A Review of Bangla Natural Language Processing Tasks and the Utility of Transformer Models. arXiv preprint arXiv:2107.03844. 2021 Jul 8.

[4] Banik N, Hafizur Rahman M, Chakraborty S, Seddiqui H, Azim. MSurvey on Text-Based Sentiment Analysis of Bengali Language. In 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT). 2019; 1-6.

[5] Iqbal T, Qureshi S. The survey: Text generation models in deep learning. Journal of King Saud University-Computer and Information Sciences. 2020 Apr 13.

[6] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I. Attention is all you need. Advances in neural information processing systems. 2017;30.

[7] Wang L, Zhao W, Jia R, Li S, Liu J. Denoising based sequence-to-sequence pre-training for text generation. arXiv preprint arXiv:1908.08206. 2019 Aug 22.

[8] Abujar S, Hasan M, Shahin MS, Hossain SA. A heuristic approach of text summarization for Bengali documentation. In2017 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT) 2017 Jul 3 (pp. 1-8). IEEE.

[9] Efat MI, Ibrahim M, Kayesh H. Automated Bangla text summarization by sentence scoring and ranking. In2013 International Conference on Informatics, Electronics and Vision (ICIEV) 2013 May 17 (pp. 1-5). IEEE.

[10] Das A, Bandyopadhyay STopic-based Bengali opinion summarization. In Coling 2010: Posters. 2010; 232–240.

[11] Akter S, Asa A, Uddin M, Hossail M, Roy S, Afjal MAn extractive text summarization technique for Bengali document (s) using K-means clustering algorithm. In 2017 IEEE International Conference on Imaging, Vision & Pattern Recognition (icIVPR). 2017; 1–6.

[12] Chowdhury RR, Nayeem MT, Mim TT, Chowdhury M, Rahman S, Jannat T. Unsupervised abstractive summarization of bengali text documents. arXiv preprint arXiv:2102.04490. 2021 Jan 26.

[13] Islam MS, Mousumi SS, Abujar S, Hossain SA. Sequence-to-sequence Bangla sentence generation with LSTM recurrent neural networks. Procedia Computer Science. 2019 Jan 1;152:51-8.

[14] Abujar S, Masum AK, Chowdhury SM, Hasan M, Hossain SA. Bengali text generation using bi-directional RNN. In2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT) 2019 Jul 6 (pp. 1-5). IEEE.

[15] Abujar S, Masum AK, Islam S, Faisal F, Hossain SA. A Bengali text generation approach in context of abstractive text summarization using rnn. InInnovations in Computer Science and Engineering 2020 (pp. 509-518). Springer, Singapore.

[16] Bangla Newspaper Dataset [Internet]. Kaggle © 2020 [cited 2021 Aug 05]. Available from https://www.kaggle.com/furcifer/bangla-newspaper-dataset

[17] Sutskever I, Vinyals O, Le QV. Sequence to sequence learning with neural networks. Advances in neural information processing systems. 2014;27.

[18] Chung J, Gulcehre C, Cho K, Bengio Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555. 2014 Dec 11.

[19] Gamallo P, Campos JR, Alegria I. A perplexity-based method for similar languages discrimination. InProceedings of the fourth workshop on NLP for similar languages, varieties and dialects (VarDial) 2017 Apr (pp. 109-114).