BanglaGPT: A Generative Pretrained Transformer-based Model for Bangla Language

Abstract—**Natural Language Processing (NLP) has entered a new era with the advent of pre-trained language models, paving the way for constructing robust language models. Pretrained transformer-based models such as GPT-2 have become prevalent due to their cutting-edge efficiency. However, these approaches rely heavily on resource-intensive languages, forcing other languages to adopt multilingual frameworks (mGPT). The mGPT model could perform better for low-resource languages such as Bangla because the model has been trained on a diverse dataset spanning multiple languages. Recent studies show that the language-specific GPT model outperforms the multilingual mGPT model. In this research, we have proposed a pretrained monolingual GPT model called BanglaGPT using the objective of causal language modeling (CLM). Due to the lack of available large datasets for NLP tasks in Bangla, we have created a Bangla language model dataset called BanglaCLM using a 26.24 GB Bangla corpus scraped from several public websites. We have used a subword-based tokenization algorithm named Byte-Pair Encoding (BPE) for Bangla and finally trained the Bangla-GPT2 model from scratch using the BanglaCLM dataset. Our pretrained BanglaGPT provides state-of-the-art performance for Bangla text generation with a perplexity score of 2.86 and a loss score of 0.45 on the test set.**

Index Terms—**Bangla NLP, BanglaGPT, Bangla Text Generation Model**

I. Introduction

Bangla is the national language of Bangladesh and a regional language of the eastern part of India. Based on the number of speakers, Bangla ranks as the seventh most spoken language around the world [1]. Moreover, a vast amount of Bangla content is generated from online news portals, emails, and social networking sites [2]. Therefore, it is required to develop the tools and techniques to process Bangla content automatically.

Automatic text generation has been one of the exciting tasks in the field of NLP due to numerous real-life applications such as auto-complete text features in word processors or mail composers. However, though a significant number of research works have been found in the literature on text generation and its application in resource-enriched languages such as English [3], minimal effort has been devoted to developing text generation tools for low-resource languages like Bangla.

With the recent advent of pretrained models [4]–[11], NLP is going through revolutionary changes for the construction of robust language models. With the pretrained models, task-specific finetuning with small datasets results from the state-of-the-art models.

In the past, sequence-to-sequence models employing Recurrent Neural Network (RNN) or Long Short-Term Memory Model (LSTM) units have been employed for text generation [12]–[14]. Recently, the transformer-based pretrained model has become the state-of-the-art model for text generation [5], [6], [9].

However, training a transformer-based model with billions of parameters is costly, considering time and computational resources. It also needs a lot of training data, which is unavailable for languages with limited resources. Therefore, a low-resource language like Bangla adopts multilingual frameworks like mGPT [15]. Multilingual models are trained on a diverse dataset spanning multiple existing languages worldwide and perform poorly on low-resource languages such as Bangla [16].

The language-specific transformer-based GPT-2 model, a descendant of the original GPT model [5], is a text generation model renowned for its ingenious efficacy. Unfortunately, no pretrained GPT-2 model for the Bangla language has been found in the literature.

The main objective of the research work is to develop a pretrained GPT-2 model for Bangla named BanglaGPT. Moreover, we deploy the BanglaGPT model for the downstream task, such as text generation.

To train the model, we have created a large Bangla corpus. We clean the dataset by filtering out the HTML tags and non-Bangla contents and apply Unicode normalization and rule-based replacements to normalize the data. We deploy a subword-based tokenization algorithm named byte-pair encoding (BPE) [17].

Finally, we train the BanglaGPT model with the preprocessed BanglaCLM dataset. The model predicts the next word from the given input text. We get a perplexity score of 2.86 and a loss of 0.45 on test dataset. Finally, we deploy the model for text generation to fill up given incomplete text.

The primary contributions of this study are

- We create a novel pretraining Bangla casual language model dataset named BanglaCLM with 26.24 GB of Bangla content.
- We train the generative pretrained transformer (GPT) based model for the Bangla language named BanglaGPT.
- Finally, we deploy the BanglaGPT model for text generation.

The remaining sections of the paper are structured as follows. Section II analyses previous studies relevant to the text generation model. In Section III, The methodology of our study is discussed. Section IV represents our results and requirement for this research work. In section V, we finally conclude the paper and provide future directions.

II. Related Works

In this section, we discuss the related previous works on the existing pretrained model for text generation tasks.

Exciting real-world applications can be found for automatic text generation in the NLP research field. Some practical applications of text generation are auto-complete feature support of modern word processors or providing text suggestions on mail composers. There are two different approaches for text generation: the RNN-based sequence-to-sequence models [12]–[14] and the transformer-based pretrained models [5], [6], [9].

A. RNN-based Sequence-to-sequence Model

In the literature, sequence-to-sequence models employing RNN or variants of RNN, such as LSTM and GRU, have been used for text generation for a long time [12]–[14]. Those models show better performance compared to Deep Neural Networks (DNNs) as the RNN-based model considers the sequential information of data which is essential for text generation. However, the RNN-based model fails to propagate sequential information for a long sequence of text which is essential for text generation [18].

B. Transformer-based Pretrained Model

With the recent introduction of pretrained models in NLP, various transformer-based pretrained models have been proposed for text generation [5], [6], [9]. The first pretrained model by utilizing the decoder architecture of the original transformer model [19] is the generative pretrained transformer (GPT) [5]. The GPT model outperforms the typical RNN-based sequence-to-sequence model because it propagates sequential information for a lengthy text sequence using the self-attention mechanism [18].

However, training a transformer-based model with many parameters is expensive. In addition, training such a transformer-based pretrained model necessitates a substantial amount of data. Due to the lack of a large dataset, low-resource language like Bangla uses multilingual frameworks such as mGPT [15] for text generation. However, multilingual models are trained on a diverse dataset spanning multiple existing languages worldwide. Therefore, such multilingual models fail to perform satisfactorily in a resource-scarce language like Bangla [16].

C. GPT-2 Model

The GPT-2 [6] model inherits the original GPT model. GPT-2 model is known vastly for its ingenious effectiveness in text generation tasks. A 40GB dataset called Web-Text, containing only English text from 8 million web pages, has been used to train the GPT-2 model using a causal language modeling (CLM) objective. The model is trained using the previous token sequence as input to predict the next probable token. Comparatively, the GPT-2 model has ten times as many parameters and training data as the original GPT model. When compared to GPT, GPT-2 has much more parameters (1.5 billion vs. 117 million). In comparison to GPT (0.4 terabytes), GPT-2 has been trained on a significantly larger corpus of text data (40 GB), enabling it to capture a wider variety of linguistic patterns and produce more varied output.

Source of data	Data size(GB)
OSCAR	12.84
Wikipedia dump	6.24
ProthomAlo	3.92
Kalerkantho	3.24

TABLE I
Sources of dataset

A larger dataset is needed for training the more recent GPT-3 model [9], which has 175 billion parameters, than its direct predecessor, the GPT-2 model. Unfortunately, such a large dataset is not available for resource-scarce languages. Therefore, we are only focusing on training the GPT-2 model.

In this research, we develop a pretrained BanglaGPT language model. Moreover, we deploy the pretrained BanglaGPT model for generating Bangla text.

III. Methodology

This section describes the data collection process, data preparation steps, as well as the BanglaGPT model architecture.

A. Dataset Collection

A huge amount of high-quality text data is necessary for pretraining large language models. For instance, BERT [4] has been pretrained using 3.3 billion tokens from the Books corpus and the English Wikipedia [20]. Later studies like RoBERTa [10] and XLNet [21] have heavily filtered and cleaned the web-crawled data. In this research work, we have created a novel dataset with 26.24 GB text to train the BanglaGPT model. We have extracted Bangla text from the OSCAR [22] multilingual crawled corpus and merged it with crawled Bangla textual data from popular websites such as Prothomalo.com, Kalerkantho and Wikipedia dump datasets. Table I shows the sources and amount of data collected from those websites.

B. Preprocessing Data

We preprocess the dataset by eliminating anomalous data, such as HTML elements and non-Bangla content. We use a Bangla normalizer library [23], which applies Unicode normalization and rule-based replacements to normalize the data.

C. Tokenizer Design

The subword-based tokenization algorithm is the most effective one for GPT-2 model. We use a popular subword-based tokenization algorithm named Byte-Pair Encoding (BPE). An illustration of the subword tokenization is given in Figure 1. The first step in BPE training is identifying the distinct set of words included in the corpus, after which the vocabulary is compiled using all the symbols used to represent those words.

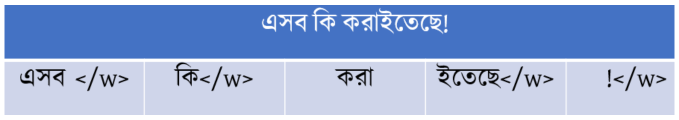

Fig. 1. Subword tokenization

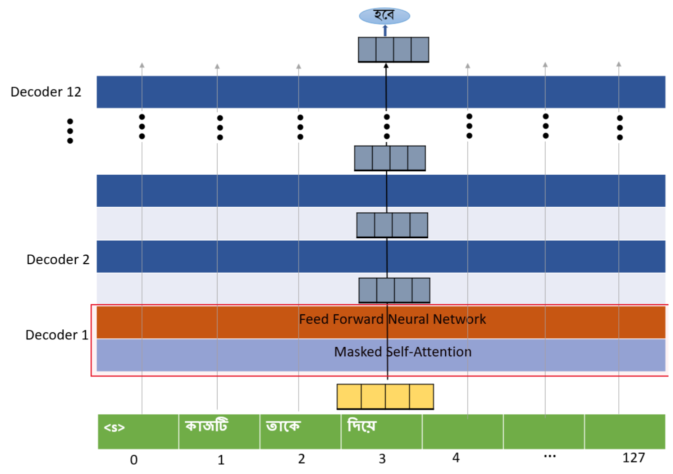

Fig. 2. Transformer architecture

Parameter Name	Value
Batch size	32
Initial learning rate	5e-5
Number of warmup steps	10000
Weight decay rate	0.01
Tokenization algorithm	BPE
Vocabulary size of tokenizer	50256
Total trainable params	124,439,808
Epochs	40
Number of training steps	40772228

TABLE II
PARAMETERS/HYPERMETERS OF THE MODEL AND TOKENIZER

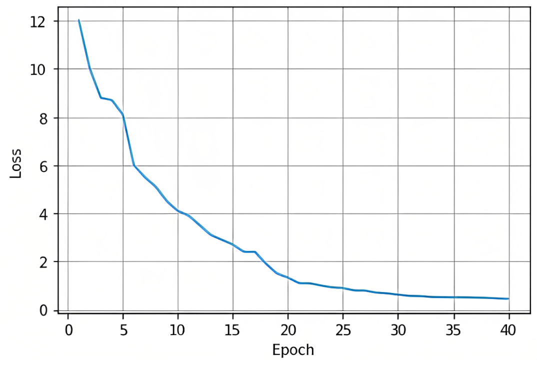

Fig. 3. Loss curve

D. Context Size and Data Collator for Language Modeling

We select a fixed context size of 128 for the tokenized input data. However, a significant portion of our dataset will be lost if the input is truncated for 128 tokens because most documents are longer than 128. To overcome this issue, we tokenize the entire input and divide it into the number of sequences using the return overflowing tokens. Finally, a HuggingFace data collator library named DataCollatorForLanguageModeling is used to generate the label data for causal language modeling (CLM). The data collator is deployed to generate the label data on the fly during training with far less memory usage.

E. Model Architecture

Figure 2 shows the GPT-2 architecture which we use for BanglaGPT model. The GPT-2 model architecture mainly employs the decoder of the original transformer model [19]. The GPT-2 model comprises a collection of decoder blocks, with each decoder block containing three crucial components: the masked self-attention block, the feed-forward neural network block, and the normalization block. The objective of the self-attention block is to determine which set of words to concentrate on. During the hidden layer, the feed-forward neural network block determines the correlation between the input words. The normalization block then normalizes the output of the feed-forward neural network.

The first decoder block sends the resulting vector up the stack to be handled by the following blocks. Finally, the output of the last decoder block is passed into linear output layers. The output tokens are chosen based on the output probabilities with the highest scores.

IV. Experiments and Results

This section describes the experimental setup and the training and evaluation results of the BanglaGPT model.

The BanglaCLM data set is divided into a training set (90%) and a validation set (10%). In addition, we have collected fresh news articles containing 4960 sentences from Prothomalo.com and Bdnews24.com to test the trained model.

Our model is trained locally on a Core i5-8600k CPU @ 3.60 GHz, 64.0 GB RAM, and Nvidia RTX 4080 16GB with Windows 10 pro-64-bit.

Perplexity (PPL) is a popular metric for evaluating language models. Perplexity is defined as a sequence of exponential average negative log-likelihood. If we have a tokenized sequence $Y = (y_0, y_1, ...y_t)$, then the perplexity of Y is,

$$PPL(Y) = exp\left\{-\frac{1}{t}\sum_{i}^{t} logp_\theta(y_i|y_{<i})\right\}$$

In the context of language models, a lower perplexity score is usually preferable.

Table II shows the hyper-parameter setting during training of the BanglaGPT model. We select 32 as the batch size. 5e-5 is specified as the initial learning rate. In order to modify the learning rate during training, the weight decay rate is adjusted to 0.01, and the number of warmup steps is increased to 10000. Each epoch contains 40772228 training steps. The total number of epochs is 40 for training the model.

After training for 40 epochs, the model has reached the point of convergence, as depicted by the loss curve in Figure

Sample input	Sample output
তাহাদের খারাপ অবস্থার কথা	তাহাদের খারাপ অবস্থার কথা চিন্তা করার কথা
আমার মতো যদি	আমার মতো যদি কোনো রকম নিবন্ধ দেখানোর চেষ্টা করে থাকেন, তা হলে কোথাও যেন

Fig. 4. Sample input/output

3. The perplexity score and loss of the trained model are 2.86 and 0.45, respectively, on the test set.

We have constructed and trained a sequence-to-sequence model using LSTM units for text generation to evaluate our trained BanglaGPT model compared to the model currently used for text generation. On the test dataset, the perplexity score of the sequence-to-sequence model using LSTM units is 10.512, and the mGPT perplexity score is 6.27, while our BanglaGPT model shows a perplexity score of 2.86. The model with a lower perplexity score performs text generation more effectively. Therefore, our proposed BanglaGPT model outperforms the sequence-to-sequence model with LSTM units and the mGPT model. Finally, we show sample-generated texts using the BanglaGPT model in Figure 4.

V. Conclusions and Future Works

Bangla is a widely spoken but resource-scarce language for which we present BanglaGPT, a monolingual generative pretrained transformer (GPT) based model. Due to the lack of available large datasets for NLP tasks in Bangla, we have created a novel Bangla text dataset named BanglaCLM with 26.24 GB of data containing Bangla text scraped from several public websites. We have used a subword-based tokenization algorithm named Byte-Pair Encoding (BPE) for Bangla and finally trained the Bangla-GPT2 model from scratch with the objective of causal language modeling (CLM) using the BanglaCLM dataset. The BanglaGPT model outperforms the mGPT and LSTM-based sequence-to-sequence model with a perplexity score of 2.86 on the test set. In the future, we will use the BanglaGPT model for downstream tasks like text summarizing and question answering. Moreover, we can extend the BanglaCLM dataset, train the GPT-3 model with the necessary hardware resources, and present a comparative study with the BanglaGPT model.

References

[1] O. Sen, M. Fuad, M. Islam, J. Rabbi, M. Hasan, M. Baz, M. Masud, M. Awal, A. A. Fime, and M. Fuad, "Bangla Natural Language Processing: A Comprehensive Review of Classical, Machine Learning, and Deep Learning Based Methods," arXiv, 2021.

[2] N. Banik, C. Saha, I. Ahmed, and K. A. Shapna, "Bangla text generation system by incorporating attention in sequence-to-sequence model," 2022.

[3] W. Yu, C. Zhu, Z. Li, Z. Hu, Q. Wang, H. Ji, and M. Jiang, "A survey of knowledge-enhanced text generation," ACM Computing Surveys (CSUR), 2022.

[4] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Association for Computational Linguistics, 2019.

[5] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," OpenAI, 2018.

[6] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," OpenAI blog, 2019.

[7] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter," arXiv, 2019.

[8] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "Albert: A lite bert for self-supervised learning of language representations," arXiv, 2019.

[9] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, and A. Askell, "Language models are few-shot learners," Advances in neural information processing systems, 2020.

[10] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," arXiv, 2019.

[11] Wang, Wentao, B. Bi, M. Yan, C. Wu, Z. Bao, J. Xia, L. Peng, and L. Si, "StructBERT: Incorporating Language Structures into Pre-training for Deep Language Understanding," arXiv, 2019.

[12] Guthaus, M. R., J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown, "MiBench: A free, commercially representative embedded benchmark suite," In Proceedings of the fourth annual IEEE international workshop on workload characterization. WWC-4, pp. 3-14. IEEE, 2001.

[13] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *Advances in neural information processing systems*, 2014.

[14] R. Nallapati, and B. Zhou and C. Gulcehre, and B. Xiang, "Abstractive text summarization using sequence-to-sequence rnns and beyond," arXiv, 2016.

[15] O. Shliazhko, A. Fenogenova, M. Tikhonova, V. Mikhailov, A. Kozlova, and T. Shavrina, "mGPT: Few-Shot Learners Go Multilingual," arXiv. Nallapati, Ramesh, B. Zhou, C. Gulcehre, B. Xiang, and others. "Abstractive text summarization using sequence-to-sequence rnns and beyond," arXiv, 2016.

[16] A. Bhattacharjee, T. Hasan, W. Ahmad Uddin, K. Mubasshir, Md. S. Islam, A. Iqbal, M S. Rahman, and R. Shahriyar, "Banglabert: Lagnuage model pretraining and benchmarks for low-resource language understanding evaluation in bangla," Findings of the North American Chapter of the Association for Computational Linguistics: NAACL, 2022.

[17] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," arXiv, 2015.

[18] J. Li, T. Tang, G. He, J. Jiang, X. Hu, P. Xie, Z. Chen, Z. Yu, W. X. Zhao, and J. Wen, "Textbox: A unified, modularized, and extensible framework for text generation," arXiv, 2021.

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

[20] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler, "Aligning Books and Movies: Towards Story-Like Visual Explanations by Watching Movies and Reading Books," IEEE International Conference on Computer Vision (ICCV), 2015.

[21] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le., "XLNet: Generalized Autoregressive Pretraining for Language Understanding," arXiv, 2019.

[22] J. Abadji, P. O. Suarez, L. Romary, and B. Sagot, "Towards a Cleaner Document-Oriented Multilingual Crawled Corpus," arXiv, 2022.

[23] T. Hasan, A. Bhattacharjee, K. Samin, M. Hasan, M. Basak, M. S. Rahman, and R. Shahriyar, "Not Low-Resource Anymore: Aligner Ensembling, Batch Filtering, and New Datasets for Bengali-English Machine Translation," In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, 2020.

[24] M. Shoeybi, M. Patwary, R. Puri, P. LeGresley, J. Casper, and B. Catanzaro, "Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism," arXiv, 2019.

[25] C. Manning, "Understanding human language: Can NLP and deep learning help?," In Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, 2016.

[26] M. Ali, N. Yousuf, S. M. Allayear, M. A. Ali, and G. Sorwar, "Generation of Bangla text from universal networking language expression,"

International Joint Conference on Advances in Signal Processing and Information Technology, Springer, 2011.

[27] N. Banik, C. Saha, I. Ahmed, and K. A. Shapna, “Bangla text generation system by incorporating attention in sequence-to-sequence model,” arXiv, 2022.

[28] O. Shliazhko, A. Fenogenova, M. Tikhonova, V. Mikhailov, A. Kozlova, and T, Shavrina. “mGPT: Few-Shot Learners Go Multilingual,” arXiv, 2022.