ISSN: 2754-6659

Journal of Artificial Intelligence & Cloud Computing

SCIENTIFIC Research and Community

Review Article

Open Access

Multi Block Transformer for Malayalam Language Modeling

Rohit TP[1*], Sasi Gopalan[2] and Varsha Shaheen[3]

[1]Department of Computer Science, SOE, Cochin University of Science and Technology, Kochi, India

[2]Department of Mathematics, Cochin University of Science and Technology, Kochi, India

[3]Department of Computer Science, SOE, Cochin University of Science and Technology, Kochi, India

ABSTRACT

In this research, we present a novel neural network architecture for natural language generation, specifically designed for Malayalam text. We have adapted the Transformer architecture which is commonly used in language modeling and extended it to work in non-Latin languages. To evaluate the effectiveness of our model, we trained it on a large corpus of Malayalam text and fine-tuned the hyper-parameters using a grid search. Our model achieved a significant improvement in generating coherent and grammatically correct Malayalam text compared to the state-of-the-art models. The model was able to generate text after just 4000 iterations and was able to effectively generalize the relation between symbols and alphabets of the language within 8000 training iterations. The transformer architecture used proved to be highly efficient in language modeling. Our work highlights the importance of developing new model architectures for text generation in complex and rich languages and opens up new avenues for future research in this area.

***Corresponding author**

Rohit TP, Department of Computer Science, SOE, Cochin University of Science and Technology, Kochi, India.

Received: February 23, 2024; **Accepted:** March 12, 2024; **Published:** March 18, 2024

Keywords: Language Modeling, Transformer Architecture, Attention Mechanism, Sequence Modeling and Transduction, Malayalam Text Generation, Text Tokenization

Introduction

The ability to generate text has become a crucial aspect of modern language processing, with applications in various fields such as machine translation, content generation, and chatbots. Despite significant progress in text generation in English, the problem remains challenging when applied to complex and rich languages such as Malayalam. Malayalam, being a South Indian language, has a large number of symbols and characters, making text generation a complex problem [1,2].

Existing solutions for text generation in Malayalam, such as recurrent neural networks (RNNs) and encoder-decoder architectures, have faced limitations in terms of computational efficiency and parallelization. The sequential nature of RNNs, which generate a sequence of hidden states by computing a function of the previous hidden state and the current input, precludes parallelization within training examples and becomes critical at longer sequence lengths. The memory constraints limit batching across examples, further adding to the computational overhead [3-6].

This research aims to address these limitations by proposing a new model architecture for text generation in Malayalam, which relies entirely on an attention mechanism to draw global dependencies between the input and output. The goal is to significantly improve computational efficiency and parallelization while achieving state-of-the-art results in terms of text generation quality [1,2].

Related Works

Attention is All You Need

This paper introduced the Transformer architecture, which revolutionized the way NLP models process sequential data. The Transformer uses self-attention mechanisms to capture dependencies between words in a sentence, without relying on recurrent connections. However, the Transformer architecture has mainly been applied to tasks and datasets in English, with limited studies in other languages. In our research paper, we aim to fill this gap by exploring the applicability of the Transformer to the processing and generation of text in Malayalam, a South Indian language. Our work extends the Transformer architecture to handle Malayalam text and demonstrates its ability to generate coherent and grammatically correct sentences in the language.

Language Models are Unsupervised Multitask Learners

In this paper, the authors showed that large language models trained on a massive amount of text data can perform well on a variety of NLP tasks without any task-specific fine-tuning. While the approach of training large language models on large datasets has proven successful in various natural language processing tasks, it may lack effectiveness when it comes to Malayalam text generation. Malayalam, being a complex language with unique linguistic characteristics, may require fine-tuning of the language model specifically for this language. The complexity of the language and the diversity of expression may pose challenges for AI in generating coherent text for longer prompts. As a result, more research and fine-tuning may be required to address these limitations in the application of language models for Malayalam text generation.

The Architecture

Overview

The architecture uses a series of Transformer blocks, each consisting of a self-attention layer and a feedforward layer. The input to each block is first passed through the self-attention layer, where the model attends to different parts of the input sequence to compute a weighted sum. The output of the self-attention layer is then passed through a feedforward layer to obtain the final output of the block. The blocks are followed by a final output projection layer, which maps the output of the blocks to the desired number of tokens in the vocabulary. The architecture is implemented using PyTorch and can be optimized using gradient descent with a learning rate scheduler.

Blocks

Each block in the architecture is designed to handle the input sequence in a parallel and efficient manner. By stacking multiple blocks, the model is able to capture more complex dependencies in the input data and improve its overall performance. Each block consists of two sub-layers: a multi-head self-attention layer and a fully connected feed-forward layer. The blocks are optimized using a step learning rate scheduler, which helps to control the learning rate of the model during training and prevent overfitting. By controlling the learning rate, the model can converge to a more accurate solution and produce better results on the task it is designed for.

Layers

Causal Self-Attention Layer

The layer performs masked self-attention, which means it only attends to elements in the input sequence that are to the left of each element in the sequence. This is achieved by using a trilinear matrix as a mask to set attention scores for elements outside of the causal window to negative infinity. The masked self-attention is followed by a linear projection to obtain the final layer output.

The input to the layer is an input sequence of shape $B \times T \times C$, where B is the batch size, T is the sequence length, and C is the number of hidden units (also referred to as the embedding dimensionality). The layer has two linear projections: the first is used to obtain the query, key, to obtain the final output after masked self-attention. The layer also contains two dropout layers for regularization.

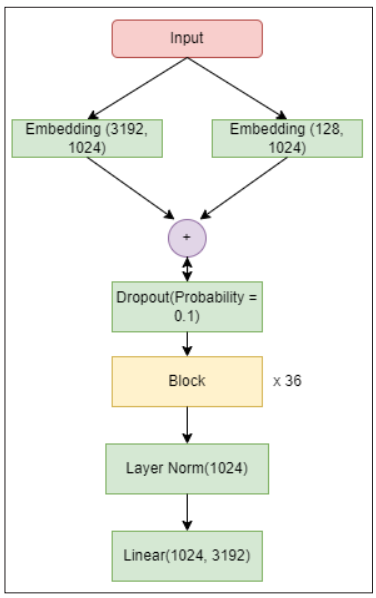

Figure 1: Layer Diagram of the Transformer Model

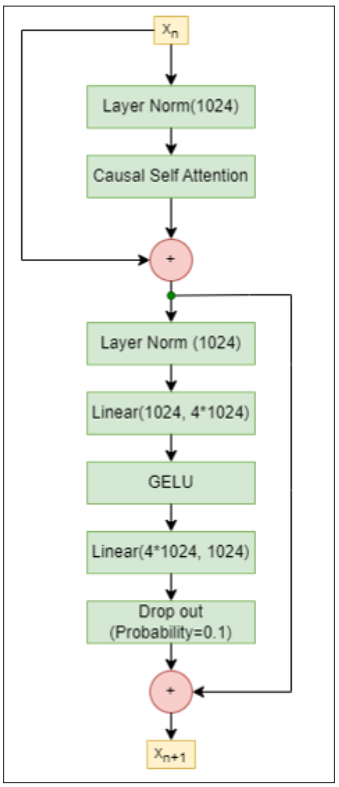

Figure 2: Expanded Diagram of Each Individual Block

The query, key, and value matrices for each head are obtained by applying the first linear projection to the input sequence. Let the output of this projection be Q, K, and V, respectively. The shape of Q, K, and V are $B \times T \times 3C$, and they are then reshaped to

$B \times n_h \times T \times \frac{C}{n_h}$, where nh is the number of heads.

The masked self-attention is performed by computing an attention score matrix A as follows:

$$A = \frac{QK^T}{\sqrt{d_k}} \qquad (1)$$

where dk is the dimension of the key vector for each head. Using a trilinear matrix, the attention score matrix is masked by setting elements outside the causal window to negative infinity. The attention scores are then normalized using the softmax function, resulting in the attention probability distribution:

$$P = \text{softmax}(A) \qquad (2)$$

The final attention-weighted representation is obtained by computing the weighted sum of the value matrix V using the attention probability distribution P:

$$Y = P \cdot V \qquad (3)$$

The final output of the layer is obtained by applying the second linear projection and applying dropout for regularization:

$$Z = \text{Dropout}(\text{Linear}(Y)) \qquad (4)$$

Feed-Forward Layer

This is a two-layer fully connected neural network with a GELU activation function in between. The layer takes an input tensor and applies a linear transformation to it, followed by the GELU

activation function, and then another linear transformation. The output of the feed-forward layer is then passed through dropout before being returned as the final result. The feed-forward layer is designed to provide a non-linearity to the input tensor and to increase the expressiveness of the model so as to increase the capacity of the model and to capture complex relationships between input and output.

$$y = \sigma(\mathbf{W}x + b) \quad (5)$$

where **W**, **b**, and $\sigma(\cdot)$ are the weights, biases, and activation functions respectively.

Training

The model was trained on a corpus of Malayalam text data collected from Wikipedia. The data consists of mostly articles and use a formal Malayalam dialect in general. The model performed self-supervised on the text.

Data Preprocessing

The data collected from Wikipedia was cleaned and any nonprintable characters were removed. To protect privacy any traceable personal information like email addresses, phone numbers, etc were replaced with mask tokens. The data set was tokenized by mapping each character to a 16-bit integer. The mapping was done using the Unicode value

Optimizer

We used AdamW optimizer with $\beta1 = 0.9$, $\beta2 = 0.95$, and learning rate = 3e-4. These specific values were derived by statistical analysis on multiple test runs and extrapolating.

Schedule

Our model was trained on a Tesla K80 GPU for 12 hrs with each training step duration of 100ms.

Result

The model was able to learn the relation between letters and symbols and also the proper use of spaces and punctuation. It was able to generate coherent completions for a given prompt as well as produce consistent outputs within its embedding size.

Table 1: Training Progress

Iteration	Train loss	Test loss
0	8.4496	8.9434
500	5.5241	5.6930
1000	3.2999	4.8662
1500	2.3141	2.8410
2000	1.3355	1.8941
2500	1.3779	4.0960
3000	1.3517	1.4768
3500	1.3013	1.7277
4000	1.2914	1.2289
4500	1.1925	1.4992

Sample Outputs

With token limit set to 36 (for short / few word completions)

- Prompt: വിദ്യാർത്ഥി
  Generated: വിദ്യാർത്ഥികളും പ്രാധാന്യം കാണുന്നു.
- Prompt: പ്രതിഷേധം
  Generated: പ്രതിഷേധം നടത്തിയിരുന്നു.
- Prompt: പ്രാദേശിക
  Generated: പ്രാദേശിക കാലാവസ്ഥയിൽ കുറവായിരിക്കും
- Prompt: പൂക്കൾ
  Generated: പൂക്കൾ വളരെ കുറവായും കാണും

With the token limit set to 108 (for long/full sentence completions)

- Prompt: ഈ പദ്ധതി
  Generated: ഈ പദ്ധതി വിദ്യാഭ്യാസ സ്ഥാപനം നടത്തുകയും അദ്ദേഹത്തിന്റെ പ്രധാന കേന്ദ്രങ്ങളെ അവതരിപ്പിക്കുകയും ചെയ്തു
- Prompt: പ്രവിശ്യയുടെ പ്രസിദ്ധമായ
  Generated: പ്രവിശ്യയുടെ പ്രസിദ്ധമായ പ്രസിദ്ധീകരണം എന്ന പുസ്തകത്തിൽ മികച്ച പല പ്രാധാന്യം കേന്ദ്രങ്ങളും കേന്ദ്ര
- Prompt: കേരളത്തിലെ
  Generated: കേരളത്തിലെ ജലസ്രോതസ്സുകളെ അടിസ്ഥാനമാക്കി ജലസംഭരണത്തിന്റെ ആത്മകഥയാണ് പാണ്ടം.

Comparison with State of the Art

The current state-of-the-art in language modeling is GPT-3 by Open AI. This model was shown to be very effective in generalizing language modeling tasks. The major drawback of the GPT models is the use of subword-level tokenization. Even though this approach is adequate in modeling Latin and Latin-based languages like English it becomes limiting when the model tries to learn languages that use complex arrangements of letters and symbols. In languages like Malayalam, the meaning is expressed using the combination of both letters and symbols and sub-word level transformers fail to learn the relation between sym- bols and letters and how the arrangement can be changed to form different words. Our architecture overcomes this problem by using a character-level model. Our model was able to learn the inter-character relation and generalize it to generate as well as infer information from unseen words. This approach opens the model to learn more in-depth characteristics of the given language and generalize.

Conclusion

In conclusion, the AI showed a remarkable capability in generating text in the Malayalam language, delivering grammatically correct outputs for short prompts with up to 36 tokens. The continuous decrease of both training and validation loss confirms the model's generalizability. However, it was observed that the AI faced difficulties in preserving context for longer text generation. Further exploration is needed to enhance the AI's coherence in generating longer sentences.

It is noteworthy that, despite the Transformer architecture's remarkable success in text generation for various languages, more efforts are required to optimize the model for the specific linguistic traits of Malayalam. Additionally, due to the intricacy of the language, the creation of large annotated datasets for training and model fine-tuning is crucial in enhancing performance.

This study provides a comprehensive understanding of the application of the Transformer architecture in generating text in the Malayalam language and highlights the significance of further research in this field. The results have the potential to be applied to a range of real-world applications, such as machine translation, text-to-speech synthesis, and text classification in Malayalam [7,8].

References

1. Ashish V, Noam S, Niki P, Jakob U, Llion J, et al. (2017) Attention Is All You Need. In Advances in Neural Information Processing Systems 5998-6008.
2. Saravanan S, Sudha K (2022) GPT-3 Powered System for Content Generation and Transformation. 2022 Fifth International Conference on Computational Intelligence and Communication Technologies (CCICT), Sonepat, India 514-519.
3. Souri A, Maazouzi Z, Achhab M, Mohajir B (2018) Arabic Text Generation Using Recurrent Neural Networks. Big Data, Cloud and Applications. BDCA 2018, Communications in Computer and Information Science 872.
4. Milanova I, Sarvanoska K, Srbinoski V, Gjoreski H (2019) Automatic Text Generation in Macedonian Using Recurrent Neural Networks. Big Data Processing and Mining, ICT Innovations, Communications in Computer and Information Science 1110.
5. Zhang X, Li Y, Peng X, Qiao X, Zhang H (2022) Correlation Encoder-Decoder Model for Text Generation. 2022 International Joint Conference on Neural Networks (IJCNN), Padua, Italy 1-7.
6. Zhou C, Shang J, Zhang J, Li Q, Hu D (2021) Topic-Attentive Encoder-Decoder with Pre-Trained Language Model for Keyphrase Generation. 2021 IEEE International Conference on Data Mining (ICDM), Auckland, New Zealand 1529-1534.
7. Radford A, Wu J, Child R, Luan D, Amodei D, et al. (2019) Language models are unsupervised multitask learners. OpenAI https://d4mucfpksywv.cloudfront.net/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.
8. Sunil R, Manohar N, Jayan V, Sulochana KG (2011) Development of Malayalam Text Generator for translation from English. 2011 Annual IEEE India Conference, Hyderabad, India 1-6.