LyricSymphony: Advancing Hindi Lyric Generation with LSTM and Transformer Models

1st Ankur Ankit
Department of Computer Science
Graduate School of Art and Science, Boston University
anankit@bu.edu

2nd Kavach Shah
Department of Computer Science
Metropolitan College, Boston University
kms22@bu.edu

***Abstract*—The study explores the application of computational tools to create Hindi poetry, aiming to merge the rich literary heritage with technology. The paper presents a comparative analysis of LSTM and Transformer models, examining their performance in generating text that captures the nuances of Hindi poetry. The results suggest that while both models learn effectively, as indicated by a converging training loss, the Transformer model shows superior qualitative performance, generating more complex, nuanced, and emotionally resonant narratives. The paper anticipates that future improvements to these models could lead to significant advancements in natural language processing and artificial intelligence, particularly in specialized domains and interdisciplinary applications.**

***Index Terms*—LSTM, Transformer, Hindi, GPT, Poetry**

I. Introduction

Poetry, an artistic form blending sound, rhythm, and meaning, has been a subject of fascination and study across various cultures and languages. In this paper, we explore the realm of Hindi poetry, written in the intricate Devanagari script, a treasure of India's rich literary heritage. Our focus is on harnessing computational tools and techniques to create a poetry generator that produces poems in Devanagari script.

Drawing inspiration from previous works in computational linguistics and poetry generation, such as the LSTM-based generation of Chinese poetry and the Prolog-based poetry generators[1], we aim to bridge the gap between traditional poetic forms and modern computational methods. Our work pays homage to the intricate rules of prosody and the aesthetic elements that define Hindi poetry, such as meter, rhyme, and figures of speech, as emphasized in past research. By adopting a computational approach, we attempt to replicate these elements in our generated poems, further enhancing the interplay between technology and the age-old art of poetry.

This project not only contributes to the field of computational linguistics but also serves as a bridge between the ancient art of Hindi poetry and the contemporary world, making this rich literary form accessible to a wider audience through technological innovation.

The paper is structured as follows: Section II Related Works. Section III dataset used for training the text generation models. In Section IV we discuss the theory, while in Section V we focus on the models implemented, results and evaluations obtained were discussed in Section VI and Concluding remarks and future scope are made in Section VII.

II. Related Works

The Machine Poetry Generator Imitating Du Fu's Styles[2]: This study represents a significant contribution to the field of automatic poetry generation. It focuses on traditional Chinese poetry, particularly the styles of the renowned poet Du Fu. Employing Long Short-Term Memory (LSTM) networks, the researchers developed a method to generate poetry that mimics specific stylistic features. Their approach involves statistical analysis of character probabilities and combinations, offering insights into author-specific styles. This work is pertinent to our project as it exemplifies the use of advanced machine learning techniques to replicate the nuances of a poet's style, a concept we adapt in handling the complexities of Hindi poetry written in Devanagari script.

Prolog Realization of a Poetry Generator[1]: This paper explores the development of a poetry generator using the Prolog programming language. It underscores the subjective nature of poetry and the challenge of defining it, a point particularly relevant to our project. The authors focus on rhythm and rhyme as key poetic elements, using rhythm templates based on existing poetry. This approach of using established poetic structures to inform computational generation is parallel to our method of incorporating the structural intricacies of Hindi poetry into our system.

Computational Linguistic Prosody [3]: Rule-Based Unified Technique for Automatic Metadata Generation for Hindi Poetry: This research delves into the computational analysis of Hindi poetry. It highlights the critical elements of Hindi poetry such as figure of speech, sentiment, and metre (Chhand), and discusses the role of prosody. The decline in traditional prosody due to lack of systematic documentation underscores the need for computational approaches like ours. This paper's focus on the structural components of Hindi poetry is directly relevant to our project, as we aim to computationally generate and analyze these components.

Design and Development of Computational Tools for Analyzing Elements of Hindi [4]: This paper presents a novel approach to analyzing Hindi poetry by mapping its key elements onto a numerical scale. It emphasizes the importance of chanda (metre), tuka (rhyme), and alankara (figure of speech) in poetry. The comparative study of these elements and their mathematical and statistical analysis provides a framework that informs our project. Their approach to quantifying poetic elements resonates with our objective to computationally generate and convert Hindi poetry while maintaining its aesthetic and emotional essence.

"Travesty Generator" by Lillian-Yvonne Bertram[5]: In this book, Bertram explores the use of algorithmic processes in poetry creation. The book is a collection of computational poems that delve into themes like the Black experience and structures of white supremacy. Bertram's approach to poetry generation involves using programming languages such as Python, and setting specific parameters for sentence structure to align with their aesthetic vision. This work can provide insights into the use of computational methods for poetic expression, particularly in terms of thematic exploration and structural design.

"A Survey on Intelligent Poetry Generation (ACL Anthology)"[6]: This paper provides a comprehensive overview of the field of intelligent poetry generation, touching upon various aspects such as targeted languages, form and content features, techniques, re-utilization of material, and evaluation. The survey covers different approaches to poetry generation, highlighting the diversity and breadth of research in this area. This resource is valuable for understanding the current state of research in computational poetry generation and the variety of techniques employed.

"The Spider's Thread: Metaphor in Mind, Brain, and Poetry" by Keith Holyoak[7]: This book, as discussed in an article in The MIT Press Reader, offers a perspective on the evolution of poetry creation, including the role of AI and computational methods. It provides a historical context for the intersection of artificial intelligence and poetry, which can be relevant for understanding the broader implications of computational poetry generation.

"Poetry at the First Steps of Artificial Intelligence" (Humanist Studies & the Digital Age)[8]: This paper discusses the emergence of poetry generation from digital humanities and its evolution into a subfield of computational creativity. It reviews the history of poetry generation, and the methods employed, and evaluates the outcomes of poetry generators. The paper also explores the relationship between computational poetry generation and human poetic movements, offering a unique perspective on the intersection of technology and creative expression.

III. Datasets

We downloaded the dataset of Hindi Poems from the Website, contains around 2500 poems the downloaded dataset link is: link

In the initial phase of our data preprocessing pipeline, we collected text data from a diverse set of HTML files, totaling 2500 documents. These files, constituting a substantial corpus, were meticulously curated for subsequent analysis. To facilitate further investigation, we amalgamated all the extracted text into a consolidated text file, a crucial step in preparing the data for subsequent processing.

The first step in refining the collected dataset involved the removal of extraneous characters that did not belong to the Devanagari script. This meticulous process ensured that the ensuing analysis would be focused exclusively on the relevant linguistic elements, enhancing the quality and coherence of the dataset.

To enhance the text's readability and maintain consistency across the dataset, we implemented a procedure to replace multiple consecutive newline characters with a single newline. Following this, we diligently stripped any leading and trailing spaces, contributing to a more uniform and standardized format for subsequent analysis.

In an effort to refine the dataset further, we executed a filtering mechanism to exclude numerical characters written in Hindi script. This step aimed to eliminate non-linguistic elements and enhance the linguistic purity of the dataset, laying the groundwork for more accurate and meaningful analyses.

The amalgamation of these preprocessing steps not only streamlined the dataset but also set the stage for a more robust and focused examination of the linguistic content within the Devanagari script. This comprehensive preprocessing pipeline not only addresses the intricacies of handling multitudinous files but also underscores our commitment to rigorously refining the dataset for subsequent research and analysis.

After Prepossessing we get the dataset which has the following information:

Dataset	
Sentences	57,038
All Words	3,58,822

IV. Theory

A. LSTM

Long Short-Term Memory (LSTM) networks[9], a special kind of Recurrent Neural Network (RNN), are pivotal in handling sequential data, making them ideal for tasks like music generation, language processing, and time-series analysis. The LSTM cell architecture is designed to overcome the vanishing gradient problem that can occur in RNNs. The vanishing gradient problem occurs when the gradients of the error function become very small as they are back-propagated through the network, making it difficult to learn long-term dependencies. The LSTM cell architecture addresses this problem by using gates to control the flow of information in the network.

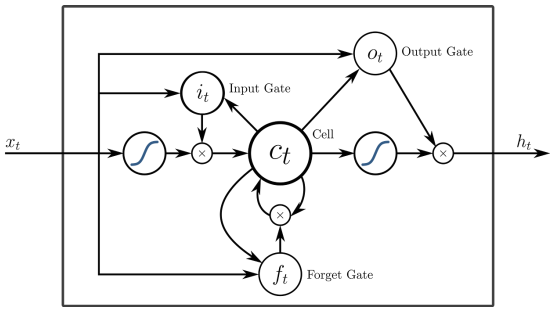

Figure 1: LSTM Cell Architecture

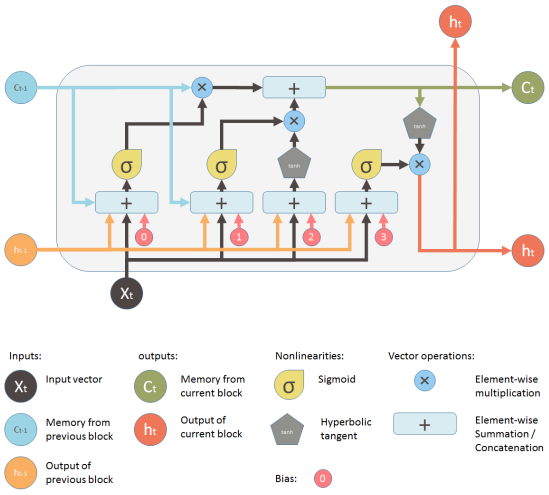

Figure 2: LSTM Model Architecture

1) Architecture of LSTM:

- **Forget Gate:** This gate decides what information to forget from the previous cell state. It takes the previous cell state (Ct-1) and the current input (Xt) as input and outputs a value between 0 and 1. A value closer to 1 means that the information will be kept, and a value closer to 0 means that the information will be forgotten.
- **Input Gate:** This gate decides what new information to store in the cell state. It takes the previous cell state (Ct-1), the current input (Xt), and a hidden state (ht-1) as input and outputs a value between 0 and 1. A value closer to 1 means that the information will be stored, and a value closer to 0 means that the information will not be stored.
- **Cell State:** This is the memory of the LSTM cell. It stores the information that is relevant to the current task. The cell state is updated based on the outputs of the forget gate and the input gate.
- **Output Gate:** This gate decides what information to output from the cell state. It takes the current cell state (Ct) and the previous hidden state (ht 1) as input and outputs a value between 0 and 1. A value closer to 1 means that the information will be output, and a value closer to 0 means that the information will not be output.

2) Problems with LSTM: The issue that occurs with RNNs in general occurs with LSTMs [10], namely that when sentences are excessively long, LSTMs still perform poorly. The reason behind this is that the likelihood of remembering the context of a word that is far away from the present word being processed drops exponentially with distance.

As a result, when phrases are lengthy, the model frequently forgets the content of distant locations in the sequence. Another issue with LSTMs is that they are difficult to parallelize because they must process sentences word by word. Furthermore, there is no concept of long and short-term dependencies. To recap, LSTMs have three issues:

- Sequential computation inhibits parallelization
- No explicit modeling of long and short-range dependencies
- Distance" between positions is linear

B. Transformer

1) Transformer Architecture Overview:: [11][12] Transformers address the challenges of parallelization in sequence transduction tasks by employing a unique architecture comprising six encoders and six decoders. These components work together to process input sequences efficiently and generate accurate translations or predictions.

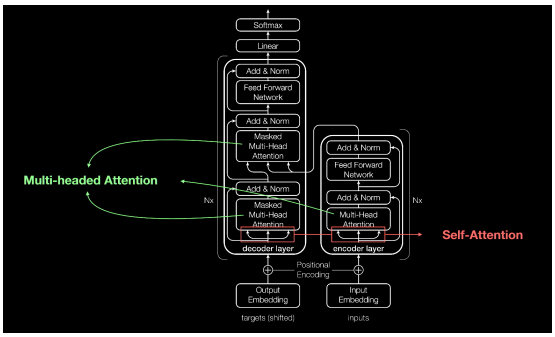

Figure 3: Transformer Model Architecture

2) Encoder Structure:: [11][12] Within each encoder, two key layers play a crucial role. First, the self-attention layer allows individual words to follow distinct paths, considering dependencies in the attention mechanism. Second, a feed-forward neural network facilitates parallel execution of these paths. The input to the encoder consists of vectors (word embeddings) of size 512, contributing to the model's ability to capture intricate relationships within the data.

3) Self-Attention Mechanism:: [11][12] The self-attention mechanism is fundamental to the Transformer's success. Each word in the input sequence is embedded into a vector of size 512, and three vectors (Query, Key, Value) are generated for each word by multiplying the embedding with matrices trained during the model's

training process. The self-attention score is computed through the dot product of the Query and Key vectors. Normalization is achieved by dividing the scores by 8 and applying a softmax operation. The final output is obtained by summing the Value vectors multiplied by the normalized scores.

4) Multihead Attention:: [11][12] To enable diverse perspectives during translation, Transformers employ multihead attention. This mechanism allows different heads to focus on various aspects of the input sequence, providing a more comprehensive representation. By leveraging multiple attention heads, the model gains the capacity to attend to different types of questions simultaneously, enhancing its translation capabilities.

5) Positional Encoding:: [11][12] Recognizing the importance of word order in sequential data, Transformers incorporate positional encoding. This additional information considers the position of each word during the encoding process, contributing to the model's understanding of the sequence's temporal dynamics. Positional encoding is crucial for preserving the sequential information that might be lost in the absence of explicit positional cues.

V. Models

A. LSTM

LSTM model starts with an Embedding layer, transforming input data into 100-dimensional dense vectors, followed by a Bidirectional LSTM layer with 150 units, enhancing learning from both forward and backward data directions. A Dropout layer with a 0.2 rate is then applied to prevent over-fitting. This is succeeded by a second LSTM layer with 100 units for processing the sequence data. The final layer is a Dense layer with *softmax* activation, outputting a probability distribution over multiple classes. The model is compiled with *categorical crossentropy* loss and the *Adam* optimizer, focusing on accuracy metrics.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 18, 100)           1859400
_________________________________________________________________
bidirectional (Bidirectional (None, 18, 300)           301200
_________________________________________________________________
dropout (Dropout)            (None, 18, 300)           0
_________________________________________________________________
lstm_1 (LSTM)                (None, 100)               160400
_________________________________________________________________
dense (Dense)                (None, 18594)             1877994
=================================================================
Total params: 4,198,994
Trainable params: 4,198,994
Non-trainable params: 0
_________________________________________________________________
```

Figure 4: LSTM Model Architecture

B. Transformer

GPT is an autoregressive language model[13] that uses the decoder part of the Transformer architecture by stacking multiple Transformer decoder layers with masked self-attention heads. This language model has been trained to predict the next word x_i given the previous words $x_1, x_2..., x_{i-1}$.[14] Its training objective is to maximize the log-likelihood where θT are the model parameters.

$$\sum_i (log(P(x_i|x_1, x_2, ..., x_{i-1})); \theta T)$$

Fig.5 shows the decoder part of the Transformer with just a single layer of Transformer block. GPT-2 small has 12 Transformer layers.

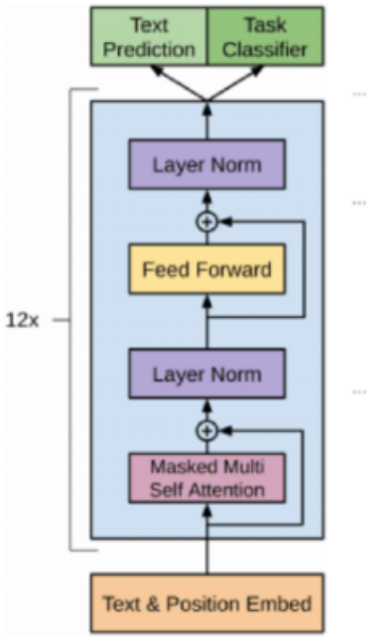

Figure 5: Decoder-only architecture used by GPT2[15]

We implement a language generation model using the GPT-2 architecture, specifically tailored for Hindi poetry. It employs the GPT2LMHeadModel and GPT2Tokenizer from the transformers library, indicating a focus on natural language processing tasks. The model is pre-trained with surajp/gpt2-hindi, which is a model that specializes in the Hindi language. The model is then fine-tuned on Hindi poetry. The training process involves custom callbacks for monitoring loss values, indicative of a detailed and iterative training approach. Additionally, the notebook includes specialized functions for generating Hindi poetry, utilizing the model's generate method with specific parameters to control the output's quality and style. Post-processing functions for text cleaning and formatting are also present, showcasing a comprehensive approach to generating stylistically coherent and contextually relevant Hindi poetry using the advanced capabilities of the GPT-2 transformer model.

VI. Results

A. Quantitative Analysis

Fig.6 is a graph showing two curves, Transformer (red) and LSTM (blue), both depicting training loss over a number of epochs during a machine learning training process. Both curves show a decrease in training loss as the number of epochs increases, which is typically a good sign of learning.

Figure 6: Comparative Training Loss Curves Over Epochs for LSTM and Transformer

To determine which one performs better, we look at which curve has the lowest loss values, especially towards the end of the training (as we approach epoch 25). The lower the loss, the better the model is performing.

However, in the graph provided, the red and blue curves are practically overlapping throughout, making it difficult to distinguish any significant difference in performance based solely on this graph. They both seem to converge to a similar value of training loss by the end of the 25 epochs. If there are any differences, they are minimal within the resolution of the graph presented.

B. Qualitative Analysis

Analysis		
Prompt	Transformer	LSTM
आकाश की ऊँचाई से	आकाश की ऊँचाई से नीचे, मेरे प्यार को जाता है। मुझको अकेला तोड़ के चल दिये, छोटी सी मुहब्बत में भी। रस्में निभा देना, हाय ग़ज़ब कहीं और न लाऊँ तुम्हें। फिर न कहना, ज़माने का डर, दोनों की महफ़िल ये नहीं हैं। किधर हो, न कैसी दुश्मनी, एक नक़्स्शएपीब फ़र इख्ती न दौराम पे रँगीर लौजी। ल न करे ली ज।	आकाश की ऊँचाई से नीचे खड़ी है तू, क्या है बुरा, क्या है कहने का हूँ दीवाना। क्या करूँ मैं? क्या करूँ मैं? क्या हुआ है? मुश्किल है प्यार का अंजाम है। दिल में है दीवाना, कौन करे कौन है मेरा। कलमा, कलमा कहाँ तेरी चाहत का दास्तां है, तू है तू ही।
तुम ही हो	तुम ही हो, मेरे दिल के फूल चुरा, तुमको खुदा की क़सम। प्यार का रंग निखर जायेगा। जिस्म को जिसने सम्भाला, वो तूफ़ान हुआ। उस से तो बेहतर है कि ज़मीन में साथ दूँ। हो देख लें आँसू बहाकर हम लोग, ऐसी बात नहीं, हमें बंदिशाएँ हैं, बग़ाहि वज़ैबीं। कैसे लूबता मुख़्बू छेड़े ना रहें, आयेंगी घटाएँ।	तुम ही हो, तुम मुझे यूँ चुराए लिए जाते हैं, मगर तुमको अजनबी से जुदा नहीं, मुझे क्या है कहना है? क्या प्यार है नहीं, क्या है काम नहीं, है क्या हां नहीं, है क्या कहना है? वो क्या है खेला तेरा, मरना जुदा होना मुश्किल है, प्यार है क्या, हुआ है क्या?
दोस्ती	दोस्ती का साथ ना छोड़ेंगे हम, क्या हुआ तेरा वफ़ा, सब कुछ है जानेजहाँ। तुझे मिलेगी निगाहों की ये सूरत, हर खुशी से हर ज़िंदगी तुझको पुकारे। आँसू आँखों में बहते हैं आशिक़ हर अपना, जो भी होता रहे बस यूँही ख्वाब देज के दिन। देते रहें हमेशा नज़र जो हँसते जुदा यूं ही रहते, चुहाएं वो मितवा बेद आए।	दोस्ती का साया नहीं है, मेरा चुनरी में है मेरा वतन भी दिल में है, मेरा प्यार न हो तो मेरा दिल जाये ना, क्यूँ जले ना, क्यूँ परवाना मेरा। नाम ना कान्हा डोलना, मैं किधर रे ओ ओ ओ ओ ओ ओ ओ ओ ओ ओ ओ ओ ओ ओ।

In a comparative analysis of the outputs generated by a Transformer and an LSTM (Long Short-Term Memory) model for three different Hindi prompts, it is evident that the Transformer model demonstrates superior performance in terms of complexity, nuance, emotional depth, and language use. For prompts such as "आकाश की ऊँचाई से" (From the height of the sky), "तुम ही हो" (It's only you), and "दोस्ती" (Friendship), the Transformer consistently produced text that was rich in imagery, emotional depth, and metaphorical language. Its responses were not only thematically consistent but also showcased a deeper understanding of contextual language generation and complex expressions. In contrast, the LSTM model, while main-

taining thematic relevance, tended to generate simpler and more straightforward text. Its responses lacked the depth and complexity of the Transformer's output but still managed to convey the basic sentiments of the prompts. This comparison highlights the advanced capabilities of Transformer models in processing and generating human-like language, surpassing the LSTM model in creating more intricate, nuanced, and emotionally resonant narratives.

VII. Conclusion and future Scope

In concluding the comparative study between Transformer and LSTM models, we observe that both models show comparable training loss over 25 epochs, as indicated by the overlapping curves in the quantitative analysis. This suggests that in terms of training efficiency, they perform similarly. However, the qualitative analysis paints a different picture, revealing the Transformer's superior capability in generating text with higher complexity, emotional depth, and richer contextual understanding. The LSTM's output, while coherent, lacks the sophistication of the Transformer model.

Looking ahead, the future scope for these models is promising. Enhancing the Transformer's architecture could further improve its performance, especially in tasks that require deep contextual comprehension. Diversifying training datasets to encompass a broader spectrum of languages and styles would likely increase both models' robustness and applicability. The Transformer's nuanced language generation shows potential for interdisciplinary applications, including computational linguistics and creative writing, suggesting that future iterations could significantly impact these fields.

Moreover, the development of hybrid models that combine the straightforwardness of LSTMs with the nuanced capabilities of Transformers could offer a balanced approach for various applications. Addressing the explainability of these models and ensuring they do not perpetuate biases are also critical future considerations. Implementing these models in real-world scenarios will provide valuable insights into their practical effectiveness and areas requiring refinement. Additionally, tailoring these models to specific domains such as legal or medical fields could cater to the demand for specialized automated language tasks.

This study underlines the rapid advancements in neural network architectures and serves as a foundation for further innovation in natural language processing and artificial intelligence, promising enhanced capabilities and applications in the near future.

A. Github

Links to the base paper, other papers related to the topic, and links to the dataset are provided in the GitHub link provided below.
Link: GitHub

References

[1] V. Atanassova, S. Pulov, "Prolog realization of a poetry generator", Proceedings First International IEEE Symposium Intelligent Systems, Varna, Bulgaria, DOI: 10.1109/IS.2002.1042586, December, 2002

[2] Kai Wang, J. Tian, Ron Gao, Chang Yao, "The machine poetry generator imitating Du Fu's styles", International Conference on Artificial Intelligence and Big Data, May 2018

[3] Milind Kumar Audichya, Jatinderkumar R. Saini, "Computational linguistic prosody rule-based unified technique for automatic metadata generation for Hindi poetry", International Conference on Advances in Information Technology, Chikmagalur, India, DOI: 10.1109/ICAIT47043.2019.8987239, February 2020

[4] Komal Naaz, Niraj Kumar Singh, "Design and Development of Computational Tools for Analyzing Elements of Hindi Poetry", IEEE Access, August 2022

[5] Lillian-Yvonne Bertram, "Travesty Generator" Book

[6] Hugo Gonçalo Oliveira, "A Survey on Intelligent Poetry Generation: Languages, Features, Techniques, Reutilisation and Evaluation", Association for Computational Linguistics Anthology, DOI:10.18653/v1/W17-3502, September 2017

[7] Keith Holyoak, "The Spider's Thread: Metaphor in Mind, Brain, and Poetry" Book.

[8] Christina Linardaki, "Poetry at the First Steps of Artificial Intelligence", Humanist Studies & the Digital Age, May 2022

[9] "Explaining Text Generation using LSTM", Blog by Abhishek Jaiswal, published on Data Science Blogathon, last updated March 2022 https://www.analyticsvidhya.com/blog/2022/02/explaining-text-generation-with-lstm/

[10] "How Transformers Work, The Neural Network used by Open AI and DeepMind", Blog by Giuliano Giacaglia, published on "Towards Data Science", March 2019 https://towardsdatascience.com/transformers-141e32e69591

[11] "Transformers in ML: What They Are and How They Work", Blog by Yulia Gavrilova, April 2023

[12] "What Is a Transformer Model?", Blog by RICK MERRITT, March 2022.

[13] Simon P Ramalepe, Thipe I Modipa and Marelie H Davel, "The Development of a Sepedi Text Generation Model Using Transformers", Southern Africa Telecommunication Networks and Applications Conference, Fancourt, Western Cape, South Africa, August 2022

[14] B. Min, H. Ross, E. Sulem, A. P. B. Veyseh, T. H. Nguyen, O. Sainz, E. Agirre, I. Heinz, and D. Roth, "Recent Advances in Natural Language Processing via Large Pre-Trained Language Models: A Survey," in arXiv preprint arXiv:2111.01243, 11 2021.

[15] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever,"Improving Language Understanding by Generative Pre-Training," 2018.

[16] "Demystifying Transformers Architecture in Machine Learning",Blog by Daivi, OCT 2023.