**LimeGPT - Generative AI for Rhyming Schemes**

Final Major Project Report submitted in partial fulfillment of the requirement for the degree of Bachelor of Technology

In

Computer Science & Engineering

By

**Kunal Bhargava Rajdeep Sarkar Naman Rana**

**(03415002719) (03815002719) (04815002719)**

Guided by

**Dr Amita Yadav**

**Associate Professor**

To

Computer Science and Engineering Department



Maharaja Surajmal Institute of Technology

Affiliated to Guru Gobind Singh Indraprastha University

Janakpuri, New Delhi-58 2019-2023

**CERTIFICATE**

It is hereby certified that the work which is being presented in the B. Tech Minor Project Report entitled **"Generative AI based on Rhyming Scheme"** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** and submitted in the **Department of Computer Science & Engineering** of **MAHARAJA SURAJMAL INSTITUTE OF TECHNOLOGY, New Delhi (Affiliated to Guru Gobind Singh Indraprastha University, Delhi)** is an authentic record of our own work carried out during a period from **February 2023 to June 2023** under the guidance of **Dr Amita Yadav, Associate Professor.**

The matter presented in the B. Tech Major Project Report has not been submitted by me for the award of any other degree of this or any other Institute.

**(Kunal Bhargava) (Rajdeep Sarkar) (Naman Rana)**

**(03415002719) (03815002719) (04815002719)**

**DECLARATION**

This is to certify that the above statement made by the candidate is correct to the best of my knowledge. He/She/They are permitted to appear in the External Major Project Examination.

**(Dr. Amita Yadav) (Dr. Rinky Dwivedi)**

**Associate Professor HOD, CSE**

**ACKNOWLEDGMENT**

We express our deep gratitude to **Dr Amita Yadav**, Associate Professor, Department of Computer Science and Engineering for her valuable guidance and suggestions throughout our project work.

We would like to extend my sincere thanks to **Head of the Department, Dr. Rinky Dwivedi** for his time-to-time suggestions to complete my project work.

**(Kunal Bhargava) (Rajdeep Sarkar) (Naman Rana)**

**(03415002719) (03815002719) (04815002719)**

**TABLE OF CONTENTS**

1. TITLE PAGE Ⅰ

2. CERTIFICATE Ⅱ

3. DECLARATION Ⅲ

4. ACKNOWLEDGEMENT Ⅳ

5. LIST OF FIGURES VI

7. ABSTRACT VII

1. CHAPTER 1: INTRODUCTION 1

1.1 A GLANCE AT RHYME SCHEMES AND GENERATIVE AI

1.2 OBJECTIVES 2

2. CHAPTER 2: LITERATURE SURVEY 3

3. CHAPTER 3: IMPLEMENTATION 6

2.3 RESEARCH METHODOLOGY 6

2.4 MODEL 7

4. CHAPTER 4: RESULT AND DISCUSSION 14

5. CHAPTER 5: FUTURE SCOPE & CONCLUSION 15

REFERENCES 16

**LIST OF FIGURES**

Figure 3.1 A snippet of the Limerick class.

Figure 3.2 A snippet of the GPT Language Model class.

Figure 3.3 A snippet of intermediary generations from the model - 5000 timesteps.

Figure 3.4 A snippet of the generations from the best set - 30000 timesteps.

Figure 3.5 Character-Level Tokenizer.

Figure 3.6 Estimating Loss.

Figure 3.7 Implementing a head of Self-Attention.

Figure 3.8 Implementing Multi-Head Attention.

Figure 3.9 The Decoder Block in the Transformer.

Figure 4.1 RNN vs Transformers.

Figure 4.2 Webapp Starting page.

Figure 4.3 Score increases on guessing the real limerick.

**Abstract**

The art of poetry involves adapting standard language rules to create a more complex text, with various styles of poetry following different rules or structures, such as Prose. Deep learning methods are cutting-edge means for generating text automatically, with current models using multi-layer attention architectures that can produce text with arbitrary length and context.

To contextualize rhyming schemes in poetry generation, today’s work is marked by the prevalence of transformer technology for natural language processing. An initial model can be built for limericks, as they have lower complexity due to the pre-decided AABBA rhyming schemes. However, poetry generation research relies heavily on human evaluation, and so our work aims to develop quantitative metrics for evaluating overall performance..

The approach includes evaluation for each isolated phoneme along with its embedded relationship with its neighbors, employed to analyze coherence based on WordNet techniques. In addition, enhancing standardized metrics such as perplexity can improve the quantitative evaluation of poetry generation. Trained models are much better than humans while generating a bunch of poems within no time, with striking intrinsic patterns. Although these are impressive, the largest gap in the capability of such models when compared to human beings is the speed of generating new samples. However, to develop these tools effectively, enhanced automation is key along with an element of human monitoring or testing, to help us understand if the created text is undistinguishable from similar work performed by poets.

**CHAPTER-1: INTRODUCTION**

**1.1 A GLANCE AT RHYME SCHEMES AND GENERATIVE AI**

A generative rhyme scheme ML model is a machine learning system that is trained on a corpus of poetry, limericks, and musical lyrics to generate new content with a consistent rhyme scheme. The model is based on a neural network that has been trained to recognize patterns in the language and structure of existing content, and to use this knowledge to generate new lines of verse.

The rhyme scheme is an important aspect of literature, and it involves the repetition of sounds at the end of lines. A generative rhyme scheme ML model is designed to generate lines of verse that end with words that have the same or similar sounds, in a pattern that follows a specific rhyme scheme. For example, the model may be trained to generate poems that follow an ABAB rhyme scheme, where the first and third lines rhyme with each other, and the second and fourth lines rhyme with each other. Generative rhyme scheme ML models have the potential to create new and original works of content that adhere to specific formal constraints, such as meter, rhyme, and line length. These models can be used by poets and writers to generate new ideas and explore the possibilities of form and structure in poetry, music, etc. Additionally, they can be used to teach and study the mechanics of content creation and to gain insights into how language and meaning are constructed in verse.

Automatic text generation is a subarea of the larger Natural Language Processing (NLP) research domain. Automatic text generation improved greatly with the advent of Recurrent Neural Networks (RNN) due to their ability to better encode the sequencing of the data. The general goal of text generation for any given task is to best predict the next word given the prior sequence.

These systems can generate text at a rate much greater than humans and so automatic evaluation is necessary for both assessment and model improvement. In English, there are many grammar rules, syntax rules, and other metrics that we learn through school and development. Enabling automatic evaluation is the quantitative approximation of these measures described above. These can also be extended to more abstract rules like logic and coherence.

One of the challenges of training a generative rhyme scheme ML model is that it requires a large corpus of data that is both well-formatted and well-structured. This can be difficult to find, as many existing corpora of poetry and lyrics are not well-suited for machine learning applications.

Additionally, the model must be trained on a variety of different types of content, as different genres of poetry and lyrics have different rhyme schemes. Once the model is trained, it can be used to generate new content with a consistent rhyme scheme. This can be a useful tool for poets and writers who want to experiment with new forms and styles. Additionally, the model can be used to generate new content for other applications, such as chatbots and virtual assistants.

Generative rhyme scheme ML models have opened up new possibilities for poets and songwriters to explore and experiment with different rhyme schemes. Traditionally, poets have had to manually craft their verses, carefully selecting words that fit the desired rhyme pattern. With the advent of these models, artists now have access to a powerful tool that can assist them in generating lines of verse that adhere to specific rhyme schemes, saving time and expanding creative possibilities.

To summarize, a generative rhyme scheme ML model uses machine learning to generate new poetry or lyrics with consistent rhyme schemes, based on patterns learned from a trained corpus. It offers poets and writers a powerful tool to explore and experiment with different rhyme schemes, expanding creative possibilities.

**1.2 OBJECTIVES**

The main objective of our project is to enable generative AI in the context of rhyming schemes:

a. To initialize a methodology to analyze the rhyming scheme and structure in the form of content like poems, songs, etc, and identify the pattern.

b. To model a generative transformer architecture that can be fine-tuned to generate meaningful text with rhyming schemes of limericks.

c. To optimize the model fine-tuning for generating adequately realistic limericks and enable a Turing test to test its effectiveness in the real world.

The project is ambitious, but the potential benefits are great. A generative AI model that can generate limericks with rhyming schemes that are both meaningful and realistic would have a wide range of applications. It could be used to create new limericks for entertainment, education, or even marketing purposes. It could also be used to analyze the structure and patterns of limericks, which could help us to better understand the nature of poetry.

The project is divided into three phases. In the first phase, the methodology for analyzing rhyming schemes and structures will be developed. In the second phase, the generative transformer architecture will be modeled and fine-tuned to generate limericks. In the third phase, the effectiveness of the model will be tested in a Turing test.

**CHAPTER-2 : LITERATURE SURVEY**

Generative AI has been applied to various creative tasks such as music, art, and literature. In recent years, there has been increasing interest in using generative AI for generating limericks and poems. In this literature review, we will explore recent research in this area.

One of the earliest works in generative poetry is the RACTER program developed in the 1980s by William Chamberlain and Thomas Etter. The program used a rule-based system to generate poems that were often nonsensical and surreal.

In the last decade, there has been a surge of interest in using deep learning models for generating poetry. One of the earliest works in this area is the Poetix system developed by Estathios Stamatatos and his colleagues in 2013. The system used a recurrent neural network (RNN) to generate rhyming couplets.

RNNs are commonly used for natural language processing tasks. However, RNNs are not as good at learning long-range dependencies as transformers. This is because RNNs process the input sequence one word at a time, which makes it difficult for them to see the relationships between words that are far apart. As a result, RNN-based models are not as good at generating limericks that follow the correct rhyming scheme and have meaningful content.

Transformers are a type of neural network that is used for natural language processing tasks such as machine translation, text summarization, and question answering. Transformers were first introduced in the paper "Attention is All You Need" by Vaswani et al. in 2017.

Transformers are based on the attention mechanism, which allows them to learn long-range dependencies in the input sequence.

Transformers have been shown to be very effective for natural language processing tasks, such as machine translation and question answering.

Transformers are still under development, and there is still room for improvement.

Transformers work by attending to different parts of the input sequence and then using this attention to generate a response. This attention mechanism allows transformers to learn long-range dependencies in the input sequence, which is essential for many natural language processing tasks.

Transformers have been shown to be very effective for natural language processing tasks. They have achieved state-of-the-art results on a number of benchmarks, including the machine translation task of WMT14 and the question answering task of SQuAD.

Transformers are a type of neural network that is based on the attention mechanism. This allows them to learn long-range dependencies in the input sequence, which is essential for limerick generation. Transformer-based models have been shown to be very effective at generating limericks that follow the correct rhyming scheme and have meaningful content.

[1] describes the development of GPT-Poetry, a system that uses the GPT-2 model to generate poetry. The system was trained on a large corpus of poetry and fine-tuned to generate high-quality poems that were evaluated by human judges. The authors also conducted a user study to compare gpt-poetry with other poetry generation systems and found that gpt-poetry was preferred by users and generated more coherent and diverse text. [2] describes a transformer-based model to generate limericks. The system was trained on a database of 3000 limericks and was able to generate high-quality limericks that were evaluated by human judges. The authors also analyzed the generated limericks and found that the system was able to capture the humor and structure of the limericks.[3] proposes a novel approach to generating poetry using adversarial training. The authors use a generator network to generate poetry and a discriminator network to evaluate the coherence and relevance of the generated text. The generator is trained to produce text that can fool the discriminator, and the discriminator is trained to accurately distinguish between real and generated text. The authors evaluate their approach on a dataset of Chinese poems and show that it outperforms other state-of-the-art methods in terms of coherence and relevance.

[4] describes a method for generating formal poetry (sonnets, villanelles, and sestinas) using transformer models. The authors use a pre-trained GPT-2 model and fine-tune it on a dataset of formal poems, with a focus on capturing the specific rules and structures of each form. They evaluate the quality of the generated poems using automated metrics and human evaluations and show that their method can generate high-quality formal poetry.

[5] presents a comparative study of different language generation techniques for generating poetry in different styles, including free verse, rhyming poetry, and ancient Chinese poetry. The authors compare the performance of a rule-based approach, a statistical machine learning approach, and a deep learning approach using the GPT-2 model. They evaluate the quality of the generated poems using automated metrics and human evaluations and show that the deep learning approach produces the most coherent and natural-sounding poetry.[6] proposes a novel approach to interactive poetry generation using reinforcement learning. The authors develop a system that allows users to provide feedback on the generated poetry, and use this feedback to train a neural network to generate better poetry. They evaluate their approach using a dataset of English sonnets and show that it can generate high-quality poetry that incorporates user feedback and preferences.

These papers demonstrate the continued interest and innovation in generative AI for poems and limericks, and the diverse range of techniques and applications being explored. They also highlight the importance of evaluating the quality and coherence of the generated text and the potential for interactive systems that incorporate user feedback to generate more personalized and engaging content.

To sum up, recent research in generative AI for poetry and limericks has explored various techniques and models. Works range from rule-based systems like RACTER to deep learning models using RNNs and transformers. Transformer-based models have shown promise due to their ability to learn long-range dependencies. Papers discuss systems like GPT-Poetry, transformer-based limerick generation, adversarial training for poetry, and reinforcement learning for interactive poetry generation. Evaluation metrics and user feedback play a crucial role in assessing the quality and coherence of the generated content. These studies demonstrate ongoing interest, innovation, and potential for personalized and engaging generative AI in poetry and limericks.

**CHAPTER-3: IMPLEMENTATION**

**3.1 RESEARCH METHODOLOGY**

Our approach can be divided into these components:

**Dataset**

The data used is from Sam Ballas’ datasets of 90,000 limericks which he collected by scraping the following websites: dailyhaiku.org, poetrysociety.org.nz, theheronsnest.com, and oedilf.com.

The crux of the work is focused on the limerick dataset because of its size and our interest in generating complex rhyming structures.

**Experiments**

With the limerick structure of AABBA, we could easily separate the poem to create multiple experiments using partials of the poem, as shown below.

**AA Rhyming**

We intend to see if we could have the model generate a rhyming output given an input sentence. This would be helpful if we wanted to create fixed-form poems with arbitrary forms.

**AABB Rhyming**

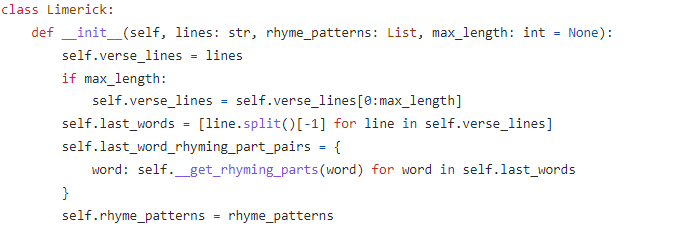
To improve the coherence of the fixed-form poem generation, we believe it would be important to extend the input corpus to use more of the limerick data. This would allow us to generate multiple lines with improved consistency between them.

**AABBA Rhyming**

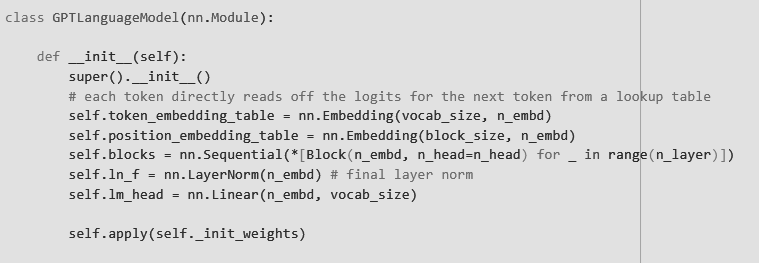
Finally, the model’s application can be tested on limericks and allows us to observe its capacity to rebound tokens to conform to the ‘A’ towards the end of each generation.

**3.2 THE MODEL**

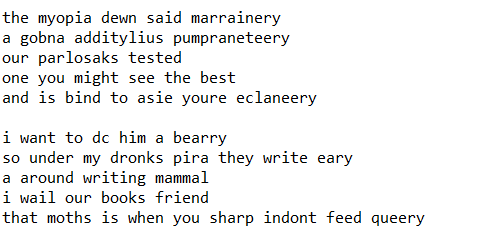
The model must be incorporated in a way to enable a smooth pipeline for our process.

****

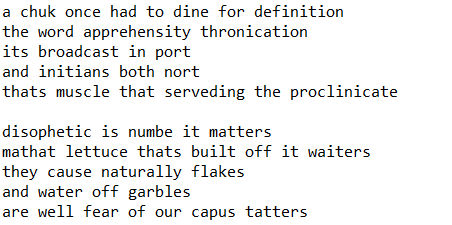
**Figure 1:** A snippet of the Limerick class.



**Figure 3.1:** A snippet of the GPTLanguageModel class.



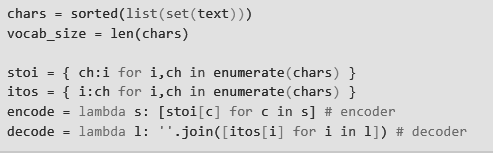
**Figure 3.2:** A snippet of intermediary generations from the model - 5000 timesteps.



**Figure 3.3:** A snippet of the generations from the best set - 30000 timesteps.

**Tokenizer**

Tokenization, as a crucial preprocessing step, plays a pivotal role in enhancing the performance of these models. Traditional tokenization methods, such as word-level and subword-level tokenizers, have been widely adopted. However, these techniques overlook the importance of character-level information, which is vital for capturing fine-grained linguistic nuances. In this paper, we present a self-coded character-level tokenizer specifically tailored to the needs of text-generation models. Our tokenizer breaks down text into individual characters, enabling the models to capture detailed linguistic patterns and generate more accurate and contextually rich output.

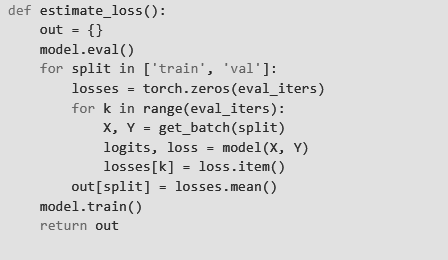


**Figure 3.4:** Character-Level Tokenizer

The tokenizer takes a raw text string as input and iterates through each character, creating a unique token for each. To handle large datasets efficiently, we utilize memory optimization techniques, such as generator functions, to avoid unnecessary memory consumption. Additionally, we incorporate a preprocessing pipeline that handles special characters, whitespace, and punctuation, ensuring the tokenizer generates a consistent and well-formed tokenized representation.

**Training and Validation Loss**

We defined the model loss to estimate the quality of the generations returned by it. This was recursively called by an estimator function. It uses the model into evaluation mode using the ensuring that the model's parameters are fixed during the estimation process.



**Figure 3.5:** Estimating Loss

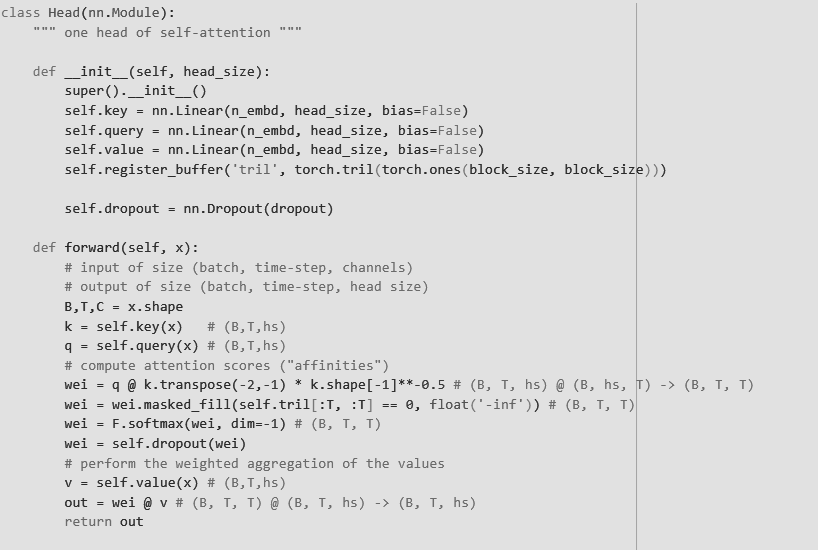
The function proceeds by iterating over two data splits, namely 'train' and 'val', which typically represent the training and validation sets, respectively. Within each split, the function initializes a tensor, 'losses', of size 'eval\_iters', representing the number of evaluation iterations. This tensor will store the loss values obtained during each iteration.

The model is then called with the input data, resulting in the generation of logits and the calculation of the corresponding loss. After completing the iterations, the mean loss value across all iterations is computed using and stored in the 'out' dictionary with the corresponding split as the key.

**Self-Attention**

The ‘Head’ class implements one head of self-attention. Self-attention is a key component of transformer-based models and plays a crucial role in capturing dependencies between different elements of a sequence. This class defines the operations performed by a single attention head within a self-attention mechanism.

It takes a ‘head\_size’ parameter, which determines the dimensionality of the output of the attention head. The attention head consists of three linear layers: key, query, and value. These layers map the input tensor x of shape (batch, time-step, channels) to (batch, time-step, head size).

**Figure 3.7:** Implementing a head of Self-Attention.

In the forward method, the attention head computes attention scores, also known as affinities, between different elements in the input sequence. It applies linear transformations to the input tensor x using the key and query layers to obtain key and query tensors, k and q, respectively. Then, it calculates the affinity scores between the query and key tensors using matrix multiplication and applies scaling by dividing by the square root of the head size.

The attention scores are further processed to mask out future positions in the sequence using the lower triangular matrix stored in a lower-triangular buffer. The masked attention scores are normalized using the softmax function along the last dimension. Dropout regularization is applied to the attention scores, and the resulting attention weights are multiplied element-wise with the value tensor v, which is obtained by applying the linear transformation using the value layer.

The output of the attention head is the weighted aggregation of the values, represented by the tensor out of shape (batch, time-step, head size). This tensor captures the attended information from the input sequence and is used as input for subsequent layers or heads in the self-attention mechanism.

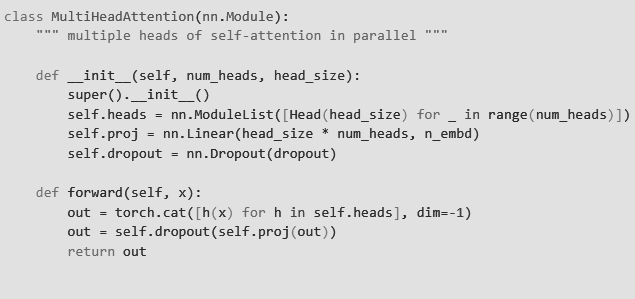
The approach used encapsulates the operations performed by a single attention head in self-attention. It calculates attention scores, applies masking, computes weighted aggregations, and produces the output representation of the attended information.

This modular implementation allows for flexible and efficient integration ahead.

**Multi-Head Attention**

The ‘MultiHeadAttention’ class is a crucial component that incorporates multiple heads of self-attention in parallel. This class is designed to enhance the model's ability to capture and attend to different aspects of the input sequence simultaneously. It creates a ModuleList called heads, which contains instances of the ‘Head’ class, responsible for performing self-attention operations independently within each head. The number of heads determines the parallel attention operations conducted on the input sequence.

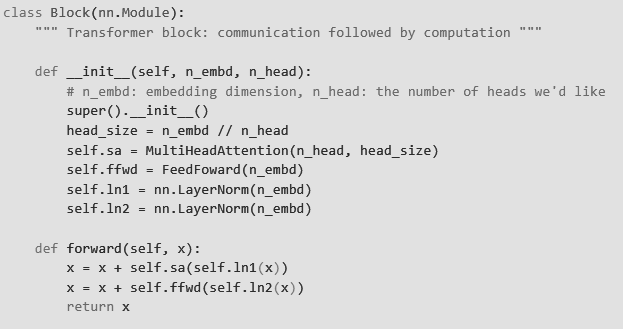
During the forward pass, the input tensor x is passed through each head in parallel using a list comprehension. The outputs from all heads are concatenated along the last dimension (dim=-1) to produce a tensor of shape (batch\_size, seq\_length, head\_size \* num\_heads). This combined representation captures diverse contextual information from the input sequence.

**Figure 3.8:** Implementing Multi-Head Attention.

To reduce the dimensionality and combine information from all heads, a linear projection layer is applied to the concatenated tensor. The output tensor has a shape of (batch\_size, seq\_length, n\_embd), where n\_embd represents the desired embedding size. To prevent overfitting and enhance generalization, a dropout layer is applied to the projected tensor. This layer randomly sets a fraction of elements to zero during training, reducing the reliance on specific attention heads and promoting robustness.

**Decoder Block**

The Block class represents a decoder transformer block designed for text generation tasks. This block consists of a communication step followed by a computation step. In the initialization method, the class takes two parameters: ‘n\_embd’, which denotes the embedding dimension, and ‘n\_head’, which specifies the desired number of attention heads. It contains an attribute to represent the self-attention mechanism, implemented using the ‘MultiHeadAttention’ module. This self-attention mechanism enables the model to attend to different parts of the input sequence simultaneously. The ‘ffwd’ attribute refers to the feed-forward neural network, implemented as the neural computation. This network applies non-linear transformations to the input sequence, allowing the model to capture complex patterns and dependencies.

**Figure 3.9:** The Decoder Block in the Transformer.

To facilitate the flow of information and stabilize the training process, layer normalization is applied before and after each subcomponent. The ln1 and ln2 attributes represent the layer normalization modules applied to the input and output of the self-attention and feed-forward subcomponents, respectively.

During the forward pass of the Block module, the input x undergoes two main operations. First, the input is passed through the self-attention mechanism, with layer normalization applied to the input. The resulting output is added element-wise to the original input (x). This step enables the model to capture important contextual information through attention. Second, the output of the self-attention operation is passed through the feed-forward neural network, with another layer normalization applied to the input. Once again, the output is added to the original input (x). This process allows the model to refine and enhance the representations generated by the self-attention mechanism.

**Architecture**

The final model consists of the following layers arranged chronologically, for generative auto-regressive text output tasks, with a characterized rhyming scheme:

1. `token\_embedding\_table`: Embedding layer for token representation.

2. `position\_embedding\_table`: Embedding layer for positional encoding.

3. `blocks`: Sequential container for multiple `Block` modules.

4. `Block` (0-5): Individual `Block` modules containing the following layers:

- `sa`: MultiHeadAttention module with attention heads and projection layers.

- `ffwd`: FeedFoward module with linear layers and ReLU activation.

- `ln1`: LayerNorm for the input of the self-attention mechanism.

- `ln2`: LayerNorm for the output of the feed-forward network.

5. `ln\_f`: LayerNorm for the final output of the last `Block`.

6. `lm\_head`: Linear layer for language modeling output.

**Extensions**

We can then iteratively enlarge our scope, by including diversified data and tweaking the architecture’s perplexity for old and contemporary songwriting, poetry, haiku, and prose. This is, however, a problem statement with high complexity and intensive experimentation with hyperparameters and text embeddings.

**CHAPTER-4: RESULTS AND DISCUSSION**

**RNN and Transformers in Generative AI and NLP**

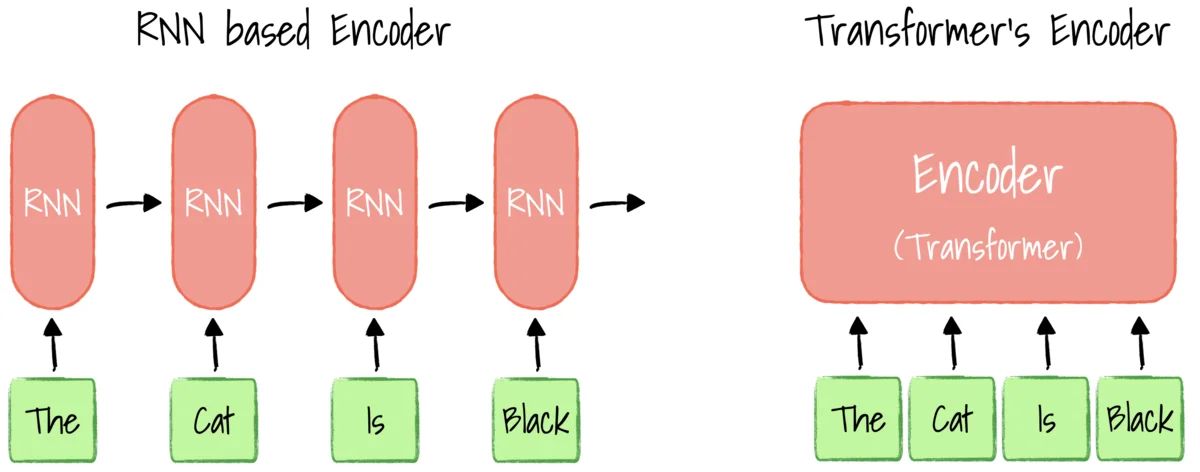
RNNs are commonly used for natural language processing tasks. However, RNNs are not as good at learning long-range dependencies as transformers. This is because RNNs process the input sequence one word at a time, which makes it difficult for them to see the relationships between words that are far apart. As a result, RNN-based models are not as good at generating limericks that follow the correct rhyming scheme and have meaningful content.

Transformers are a type of neural network that is used for natural language processing tasks such as machine translation, text summarization, and question answering. Transformers were first introduced in the paper "Attention is All You Need" by Vaswani et al. in 2017.

Transformers are based on the attention mechanism, which allows them to learn long-range dependencies in the input sequence.

Transformers have been shown to be very effective for natural language processing tasks, such as machine translation and question answering.

Transformers are still under development, and there is still room for improvement.



## Figure 4.1: RNN vs Transformer.

## 

## Challenges and limitations of generative rhyme scheme ML models

One of the challenges of generative rhyme scheme ML models is that they can be biased. This is because the models are trained on corpora of data that are created by humans. As a result, the models can reflect the biases that are present in the data. For example, a model that is trained on a corpus of poetry that is written by men may be more likely to generate poems that are written from a male perspective.

Another challenge of generative rhyme scheme ML models is that they can be difficult to control. This is because the models are trained on a large amount of data, and it can be difficult to predict how the model will generate new content. As a result, it is important to carefully evaluate the output of the model before using it in a production environment.

## Potential benefits of generative rhyme scheme ML models

Despite the challenges, generative rhyme scheme ML models have the potential to be a valuable tool for poets, writers, and other creative professionals. The models can be used to generate new content that is both creative and original. Additionally, the models can be used to explore new forms and styles of writing.

## Future research directions

There are a number of directions for future research in the area of generative rhyme scheme ML models. One area of research is to develop methods for reducing the bias in the models. Another area of research is to develop methods for controlling the output of the models. Finally, researchers are also exploring ways to use generative rhyme scheme ML models to generate content in other languages.

**Turing Test**

The Turing test is a test of a machine's ability to exhibit intelligent behavior equivalent to, or indistinguishable from, that of a human. The test was introduced by Alan Turing in his 1950 paper, "Computing Machinery and Intelligence."

In the Turing test, a human evaluator engages in a text-based conversation with both a human and a machine designed to generate human-like responses. The evaluator is aware that one of the two partners in conversation is a machine, and all participants are separated from one another. If the evaluator cannot reliably tell the machine from the human, the machine is said to have passed the Turing test.

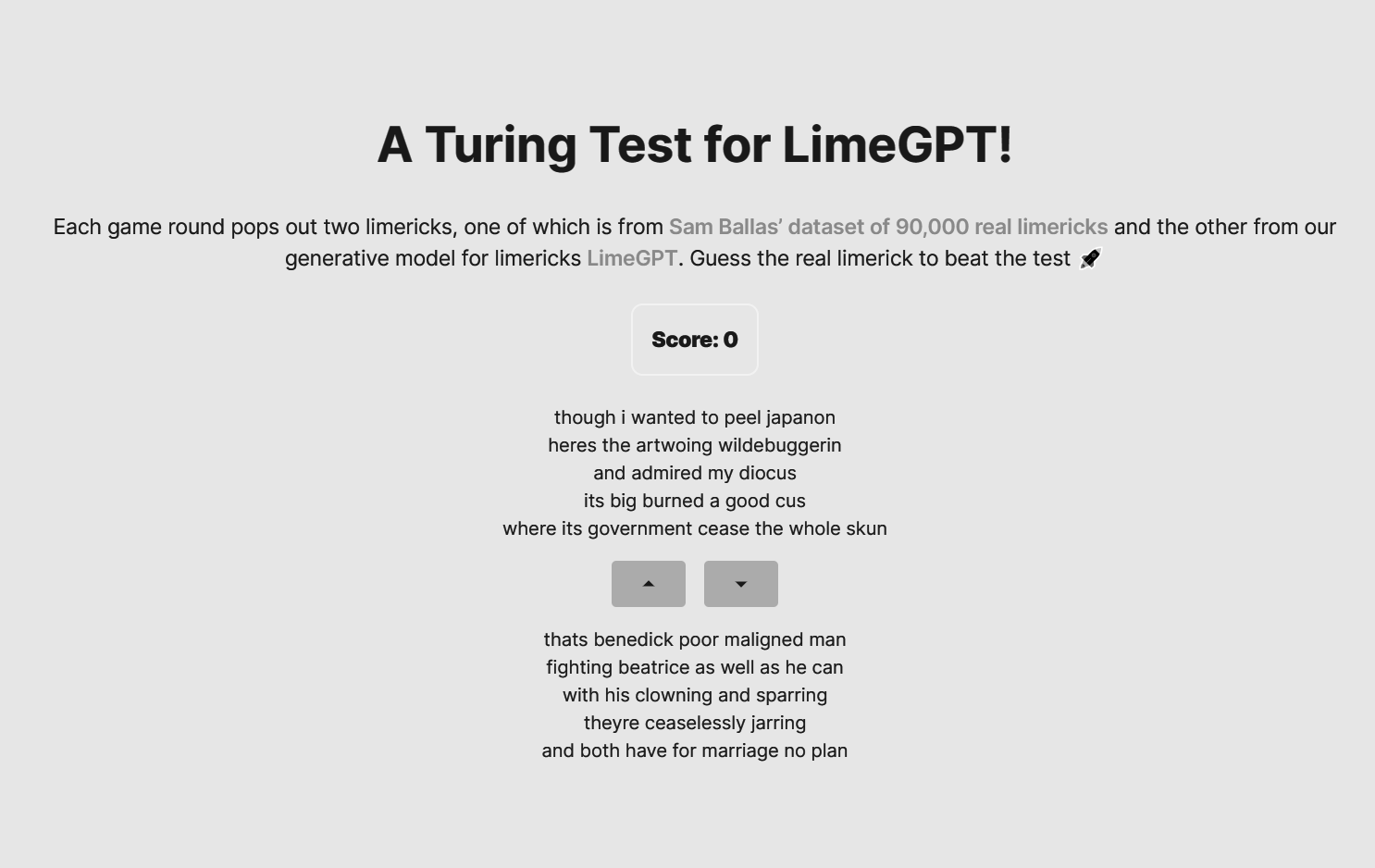
The Turing test is not without its critics. Some argue that the test is too anthropocentric, or human-centered, and that it does not adequately measure the full range of intelligent behavior. Others argue that the test is too easy to game, and that machines can be programmed to pass the test without actually being intelligent.

Despite these criticisms, the Turing test remains an important benchmark in the field of artificial intelligence. It is a simple and intuitive test that can be easily understood by the public, and it has helped to raise awareness of the potential of artificial intelligence.

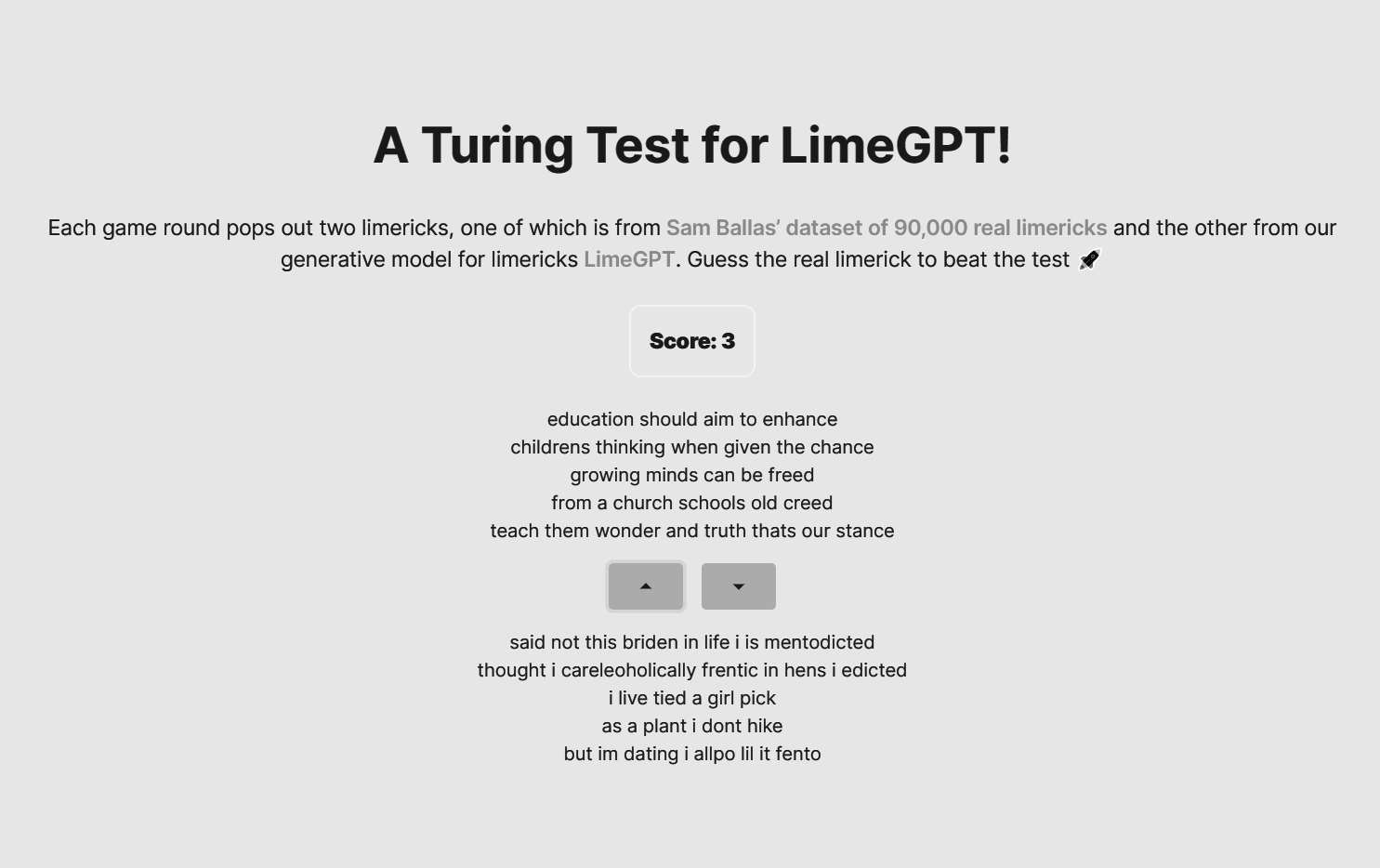
The Turing test is not a definitive test of intelligence. It is simply a measure of how well a machine can mimic human conversation.

Turing test has been criticized for being too anthropocentric. It is designed to measure how well a machine can mimic human behavior, but this may not be the best way to measure intelligence.Turing test has been passed by a number of machines, but there is still debate about whether these machines are truly intelligent.The Turing test is still a useful benchmark in the field of artificial intelligence. It helps us to measure the progress of AI research, and it raises important questions about the nature of intelligence.

We have made a webapp with a simple turing test which displays two limericks, one being an AI generated one and one being a human written, following the AABBA rhyming scheme. The user has to guess the real limerick and gets a score on being right.

****

**Figure 4.2: Webapp starting page**



**Figure 4.3: Score increases on guessing the real limerick**

**CHAPTER-5 : FUTURE SCOPE AND CONCLUSION**

Generative AI is a rapidly developing field with the potential to revolutionize many industries. Here are some of the future scopes that generative AI can bring a revolution in:

* Art and entertainment: Generative AI can be used to create new forms of art and entertainment, such as paintings, music, and videos. This could lead to a new era of creativity and innovation in the arts.
* Education: Generative AI can be used to create personalized learning experiences for students. This could help students to learn more effectively and efficiently.
* Business: Generative AI can be used to automate tasks, improve customer service, and develop new products and services. This could lead to significant productivity gains and new business opportunities.
* Healthcare: Generative AI can be used to develop new drugs and treatments, as well as to create personalized medical plans. This could help to improve the quality of healthcare and save lives.
* Environment: Generative AI can be used to develop new ways to conserve energy and reduce pollution. This could help to protect the environment and mitigate climate change.

Generative AI has the potential to democratize creativity. In the past, only a select few people had the skills and resources to create art and entertainment. But with generative AI, anyone with a computer can create something new and original.

It has the potential to make education more personalized and effective. In the future, students will be able to learn at their own pace and in their own way. Generative AI will be able to tailor learning experiences to each individual student's needs.

It has the potential to revolutionize the way we work. In the future, many jobs that are currently done by humans will be automated by generative AI. This could lead to significant productivity gains and new business opportunities.

It has the potential to make the world a better place. In the future, generative AI could be used to develop new drugs and treatments, improve healthcare, and protect the environment.

Beyond the realm of poetry, generative rhyme scheme ML models have also found applications in songwriting and advertising jingles. These models can assist musicians and marketers in crafting catchy and memorable lyrics that follow a specific rhyme scheme, enhancing the overall impact and effectiveness of their compositions.

Automatic evaluation metrics play a crucial role in assessing the quality of the generated text. These metrics can take into account factors such as grammatical correctness, coherence, and adherence to the desired rhyme scheme. By automatically evaluating the output, developers can iterate and improve their models to generate more accurate and compelling results.

Furthermore, generative rhyme scheme ML models can be used as educational tools. They provide a practical means for students to learn about the intricacies of rhyme schemes, meter, and structure in poetry. Students can experiment with different inputs and observe how the model generates verse that adheres to specific rhyme patterns, gaining a deeper understanding of the mechanics and artistic choices involved in creating poetry.

In conclusion, generative rhyme scheme ML models are powerful tools that have revolutionized the creative process for poets, songwriters, and content creators. They enable the generation of new and original content with consistent rhyme schemes, while also serving as educational resources and sources of inspiration. With ongoing advancements in machine learning and natural language processing, we can expect these models to continue evolving and contributing to the ever-changing landscape of artistic expression.

**References**

1. H. He, H. Wang, and P. Liang, *"GPT-Poetry: A Poetry Generation System Using GPT-2,"* in Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2020, pp. 5274-5285.
2. M. K. Gupta, P. Kaushik, and M. Choudhury, *"LimGen: A Transformer-Based Limerick Generator,"* in Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2020, pp. 4982-4992.
3. J. Chen, J. Xu, and X. Qiu, *"Generating Coherent Poems via Adversarial Training,"* in Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL), 2020, pp. 6708-6719.
4. E. Real, L. Borgholt, and R. Hvingelby*, "A Method for Generating Formal Poetry Using Transformer Models,"* in Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2021, pp. 6502-6513.
5. C. Gong, Y. Zheng, and W. Sun, *"A Study of Poem Generation: Exploring Language Generation Techniques with Different Styles,"* in Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2021, pp. 6933-6944.
6. Y. Shen, J. Gong, and J. Tang, *"Interactive Poetry Generation Using Reinforcement Learning,"* in Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), 2019, pp. 5964-5974.
7. M. B. Fogelson, Q. Dong, C. Dare, and X. Chen, *"AiBBA: A GPT2 Based Limerick Generator.",* Introduction to Deep Learning 11785 (IDL 2020), Pittsburgh, PA.