PoetAI: Automatic Limerick Generator

Aditya Bindra
Heinz College
Carnegie Mellon University
Pittsburgh, PA 15213
adityabi@andrew.cmu.edu

Aditya Malani
Heinz College
Carnegie Mellon University
Pittsburgh, PA 15213
amalani2@andrew.cmu.edu

Bandish Parikh
Heinz College
Carnegie Mellon University
Pittsburgh, PA 15213
bparikh@andrew.cmu.edu

Nirmalsing Patil
Heinz College
Carnegie Mellon University
Pittsburgh, PA 15213
nirmalsp@andrew.cmu.edu

Rita Singh
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
rsingh@andrew.cmu.edu

Yashash Gaurav
Heinz College
Carnegie Mellon University
Pittsburgh, PA 15213
ygaurav@andrew.cmu.edu

Abstract

For the longest time, machine learning models have been deployed to perform analytical tasks and identify patterns in data that are otherwise difficult to discern by the human eye. These tasks are highly objective in nature and have a fixed "true" value that we want the machine to be able to predict. However, what if we could use 'art'-ificial intelligence to create art? The expected output here wouldn't be a numerical value or a class, but rather an art piece that is creative, expressive, and appealing to humans. For our project, we focus on using machine learning to generate poetry. Specifically, PoetAI generates limericks (5-line contextual poems with AABBA rhyme scheme) of high quality where quality is assessed using 2 metrics - rhyming and context. We are proposing a pipeline driven by transformer-based models that generates limericks and then scores and filters the good quality limericks based on various aspects including rhyme and context. A final language model tries to fix the rhyming to enhance limerick quality. PoetAI holds numerous business applications, especially in the field of customized marketing and messaging.

1 Introduction

Poetry is an outcome of creation. It is a form of literary work that is often characterized by an interplay of thought-provoking words meant to stimulate the human mind. A limerick is a short humorous form of verse that has five lines and follows the AABBA rhyme scheme. What if we could use 'art'-ificial intelligence to create art? While a lot of research has been done in the field of Natural Language Understanding, the area pertaining to generation and qualitative analysis of poetry still remains to be explored. Even after much research, AI has been criticized for lacking creativity and for occasionally producing texts that make no sense. The expected output in this case isn't a simple numerical value or a class label, but rather an art piece that is meant to be creative, expressive, and appealing to humans. We derived that, in the context of an Artificial Intelligence algorithm, creativity is just the development of clearly stated mathematical objective functions that a model must be optimized on. The desired output of creativity can not be captured by conventional loss functions. Our main objective is to respond to the question, "What makes this piece of poetry/limerick a good one?" while giving objective functions to grade a specific piece of poetry/limerick. Will a large language model be able to learn the art of poetry? In this project we used 4 models as described in the flow section, GPT2 for limerick generation, LSTM based rhyme scorer for evaluating the rhyming

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

scheme of the limerick, pretrained sentence transformer model MiniLM-L6-V2 for evaluating context of the limericks and Masking BERT for fixing the rhyme of a limerick with a good context score. This paper starts with Literature review section followed by methodology which contains the details about the architecture of all four models used. All the experiments are listed in Training and Experiment section.

2 Literature Review

2.1 GPoet-2 : GPT2 based transformer with forward and reverse fine tuning

Another work that we referenced is GPoeT-2: A GPT-2 Based Poem Generator [6], where the authors propose a two-stage free-form limerick generation. The proposed two-stage generation uses the forward language model to generate a limerick's first line with high quality and diversity, then uses the reverse language model to generate the rest of the four lines given the first line generated by forwarding LM. They also select and evaluate a few metrics that quantify the idea of "good poetry" such as syntactical correctness, lexical diversity, and subject/topic consistency.

2.2 LimGen : GPT2 based limerick generator which uses Search Metrics to enforce rhyme

One of the recent works in this space is another GPT-2 implementation called LimGen by Jianyou Wang et al.[8] where they use search metrics such as the Adaptive Multi-Templated Constraint algorithm that constrains their search to the space of realistic poems, the Multi-Templated Beam Search algorithm which searches efficiently through the space, and the probabilistic story-line algorithm that aims to provide coherent story-lines related to a user-provided prompt word.

2.3 Deep-Speare : RNN based Sonnet generator with rhyming dictionary to enforce rhyming

Deep-speare[17] is a Sonnet based model used to capture language, rhyming and meter of poetry. These models under-performed in generating human level poetry but served as good reference for rhyme capture with models. Rhyme was enforced by a cosine similarity of the last words generated by the model and a loss function was employed to penalize model when it was not rhyming. A rhyming dictionary was maintained to pick words based on the context.

3 Methodology

3.1 Process Flow

The Baseline architecture we started with performed well but not enough to be able to label the model a good poet. Hence we reworked the architecture and also added more datasets for training. We also changed the loss, and the way loss is propagated. Hence in addition to retraining GPT-2, we were able to come up with a scoring and filter-and-fix mechanism that makes use of a rhyme scorer and a context scorer to first assess the quality of Limericks and subsequently pass them through RhymeBERT in order to re-sample words for limericks with good context but poor rhyming.

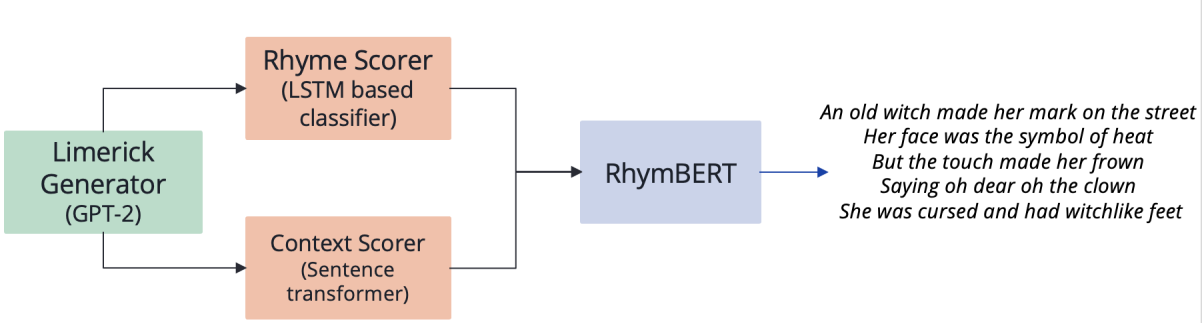

Figure 1: New Architecture

Our Limerick generator is a large language model GPT2 with 345M parameters. This was fine tuned on limerick dataset which consisted of 150k+ limericks. Architecture for training is same as baseline model GPT2.

3.2 Model Description

3.2.1 Limerick Generator GPT2

We implemented two baseline models, GPT2 and OPT350 to compare how each model performed on the task of Limmerick Generation. Our first base model was based on the work done by the PoetAI team [4] to generate limericks. Their approach was inspired by the work done by Tuan Nguyen et al [5], where the PoetAI team used GPT-2 followed by a self-attention LSTM to generate context vectors for the 5 lines of limericks.

We used masked self-attention heads to train a 12-layer decoder only transformer (768 dimensional states and 12 attention heads).

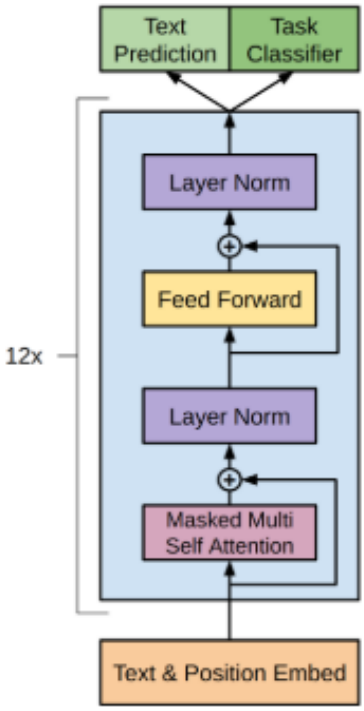

Figure 2: GPT2 Architecture [15]

Tokenizer: We used standard GPT2 pretrained tokenizer (medium) with the maximum length capped at 64.
Optimizer: We used the Adam optimization scheme with a max learning rate of 1e-4.
Scheduler: We used Linear Scheduler with warmup to increase the learning from a small learning rate to a constant with a warmup step of 10000.

We trained it for 160,000 iterations on approximately 148,000 data points and the final training loss was 2.338. Following are some examples of limericks generated by the model. The number of such good limericks with perfect rhyme-scheme (AABBA) is approximately 200 out of every 1000. This is an improvement over the previous work carried out and we further propose a filter-and-fix pipeline with the aim of identifying limericks with good context but that don't adhere to the AABBA rhyme in an attempt to try and 'fix' their rhyming using a RhymeBERT model. You can refer the code at : https://github.com/nirmalpatil/PoetAI/tree/main

An old witch made her mark on the street,
her face was the symbol of heat
but the touch made her frown,
saying oh dear oh the clown
she was cursed and had witchlike feet.

I just went for a swim in the bay,
with my swimwear so light I should say
that i just couldn't cope,
with my cinchons as dope
I'd say I should have stayed home today.

3.2.2 Rhyme Scorer

The baseline model fell short in enforcing the rhyme scheme as required by the limerick design (AABBA). Below is an example generated by the model:

When asked for some detail it's clear
It's the claim they must be clear
That this date is too long
But I cant understand why
So I tried my best not have you see

On looking at more examples generated by the baseline model, we observed that in its current form, the model is not doing a perfect job of generating AABBA rhyme scheme always. As a way to quantify this, we use a discriminator (Siamese LSTM) to measure rhyming[9].

Essentially, discriminator is a classifier that takes in a pair of words as inputs and produces a binary output 1/-1 based on whether the two words are rhyming or non-rhyming respectively. For our use case, we have used a Siamese LSTM[10] based architecture as described in Figure 1.

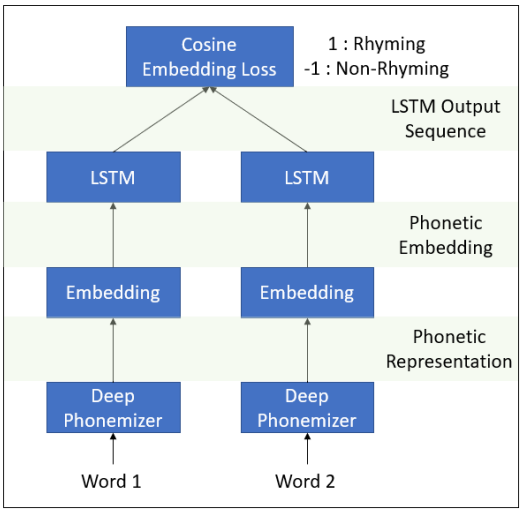

Figure 3: Discriminator Architecture

The pair of rhyming words are first transformed into phonemes using DeepPhonemizer model[11]. These phonemes are then converted into embeddings using an embedding layer. These embeddings are then put through a shared parameter bidirectional LSTM network. The output tensors from these LSTM blocks are then fed into Cosine Embedding Loss that works on cosine similarity based similarity matching. You can refer the code at : https://github.com/nirmalpatil/PoetAI/tree/main

3.2.3 Context Scorer

For context scoring, we use the all-MiniLM-L6-V2 model which has been fine-tuned on a 1B sentence pairs dataset. It uses a contrastive learning objective: given a sentence from the pair, the model tries

to predict which out of a set of randomly sampled other sentences, was actually paired with it in the dataset. It maps sentences paragraphs to a 384-dimensional dense vector space [18] which we further use to find semantic similarity using Cosine Similarity.

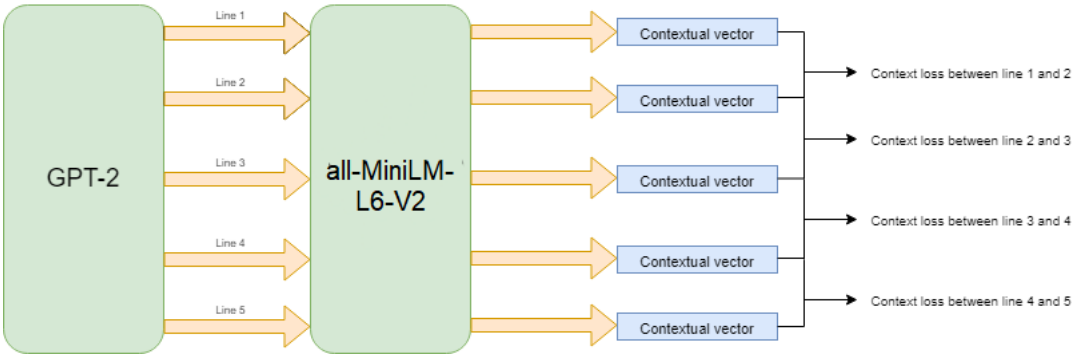

Figure 4: Context Calculation

3.2.4 RhymBERT

BERT, short for Bidirectional Encoder Representations from Transformers, is a transformer-based machine learning model for natural language processing. BERT was specifically trained on Wikipedia and Google's BooksCorpus(approximately 800M words). These large informational datasets contributed to BERT's deep knowledge[20]. BERT has been trained on two prediction tasks: masked language modeling (MLM) and next-sentence prediction(NSP). Figure 1 illustrates the original BERT architecture for MLM where a sequence with masked tokens is fed to a transformer encoder that generates output vectors. These vectors are then passed through a fully connected classification layer and a softmax that assigns probability at each step in the sequence.

The cross-entropy loss ($-\sum_{c=1}^{M} y_{o,c} \log(p_{o,c})$) function only considers the prediction of masked values and ignores everything else.

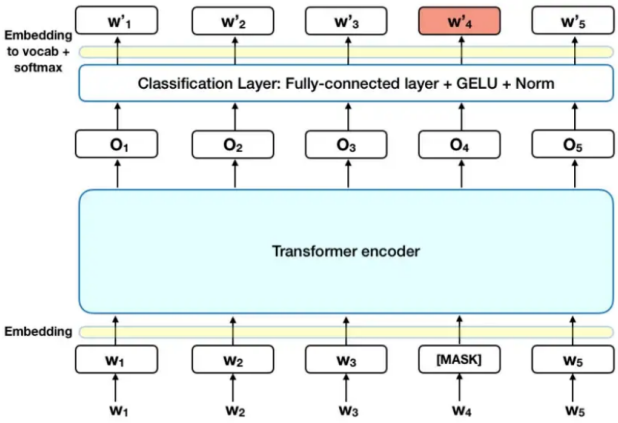

Figure 5: BERT MLM Architecture [23]

For our use case, we fine-tune the pre-trained BERT model on the masked language model. We make use of this MLM task of BERT to enhance/fix the rhyming of the generated limericks. We utilized the limerick training dataset and masked the last words of all five sentences in the limericks. BERT's bidirectional learning from the text by masking (hiding) a word in a sentence and forcing BERT to bi-directionally use the words on either side of the covered word to predict the masked word helps in learning the rhyming pattern. Below is an example of the training samples fed for fine-tuning:

Some folks get a kick out of [MASK]
With jokes that fool people they're [MASK]

Of course, they are [MASK]
Since the people they [MASK]
With their jokes are not usually [MASK]

The objective during the inference phase would be that the limericks that meet the contextual threshold but lack rhyming are fed to this fine-tuned model[19]. It would then replace the last word in the limerick to enforce rhyming. You can refer the code at :https://github.com/nirmalpatil/PoetAI/tree/main

3.3 Dataset Description

3.3.1 Limerick Dataset

We merged two datasets for training the model. First is Omnificent English Dictionary In Limerick Form (OEDILF) [21] with 90k Limericks and and 70k limericks from Zenodo dataset [22]. All these limericks were downloaded from the respective source in text file. Limericks containing more than 5 lines were removed. All the punctuation marks were removed before feeding it GPT2. Limericks containing expletive words were dropped. Dataset was clean enough so no further processing was required.

3.3.2 Rhyme Scorer Dataset

We extracted the pairs of rhyming and non rhyming words from the above merged dataset of 160k limericks. 150k+ rhyming and non rhyming pairs were extracted by focusing on the last words for lines adhering to AABBA pattern. Label 1 was added if the words are rhyming ; AA, BB else 0. for e.g. (word1 : toy, word2: joy, label :1),(word1 : toy, word2: peace, label :0)

4 Training, Experiments and Results

We implemented two baseline models, GPT2 and OPT350 to compare how each model performed on the task of Limmerick Generation. Our first base model was based on the work done by the PoetAI team [4] to generate limericks. Their approach was inspired by the work done by Tuan Nguyen et al [5], where the PoetAI team used GPT-2 followed by a self-attention LSTM to generate context vectors for the 5 lines of limericks.

4.1 Limerick Generator

4.1.1 GPT2 345M

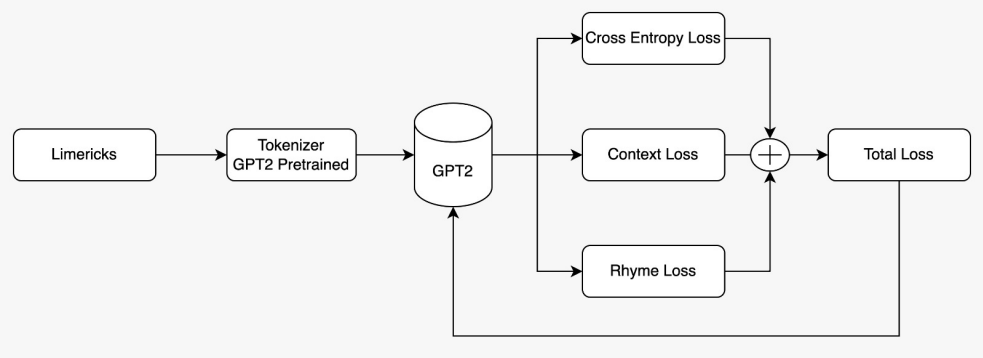

Figure 6: Model Architecture

Our baseline model closely mimics the first transformer design. We used masked self-attention heads to train a 12-layer decoder only transformer (768 dimensional states and 12 attention heads).

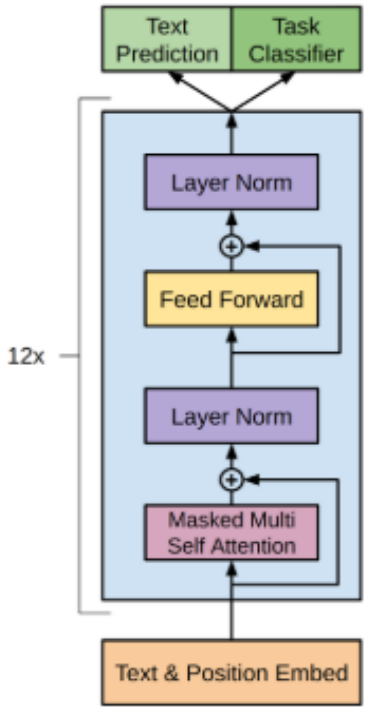

Figure 7: GPT2 Architecture [15]

Tokenizer: We used standard GPT2 pretrained tokenizer (medium) with the maximum length capped at 64.
Optimizer: We used the Adam optimization scheme with a max learning rate of 1e-4.
Scheduler: We used Linear Scheduler with warmup to increase the learning from a small learning rate to a constant with a warmup step of 10000.

We trained it for 160,000 iterations on approximately 148,000 data points and the final training loss was 2.338. Following are some examples of limericks generated by the model. The number of such good limericks with perfect rhyme-scheme (AABBA) is approximately 200 out of every 1000. This is an improvement over the previous work carried out and we further propose a filter-and-fix pipeline with the aim of identifying limericks with good context but that don't adhere to the AABBA rhyme in an attempt to try and 'fix' their rhyming using a RhymeBERT model.

An old witch made her mark on the street,
her face was the symbol of heat
but the touch made her frown,
saying oh dear oh the clown
she was cursed and had witchlike feet.

I just went for a swim in the bay,
with my swimwear so light I should say
that i just couldn't cope,
with my cinchons as dope
I'd say I should have stayed home today.

4.1.2 OPT 350M

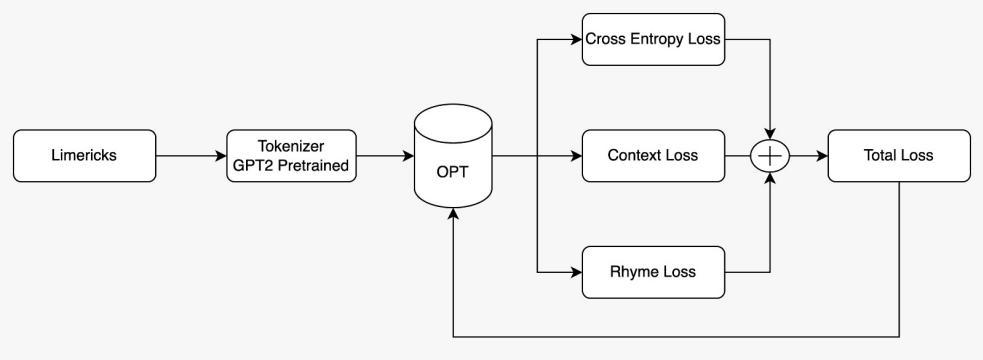

Figure 8: Model Architecture

Our baseline model2 (OPT) closely mimics the design. We kept everything same as the previous architecture just changing the model to OPT(350M).

Motivation : Recently, Meta AI published "OPT: Open Pre-Trained Transformer Language Models" and an associated code repository with the intent of open-sourcing high-performing large language models (LLMs) to the public. Despite the fact that LLMs have demonstrated impressive performance on numerous tasks (e.g., zero and few-shot learning), they have only been made available to the public via APIs. Such a paradigm is problematic from a research perspective, and hence we decided to try out an open source LLM which has equivalent performance to that of GPTs [14].

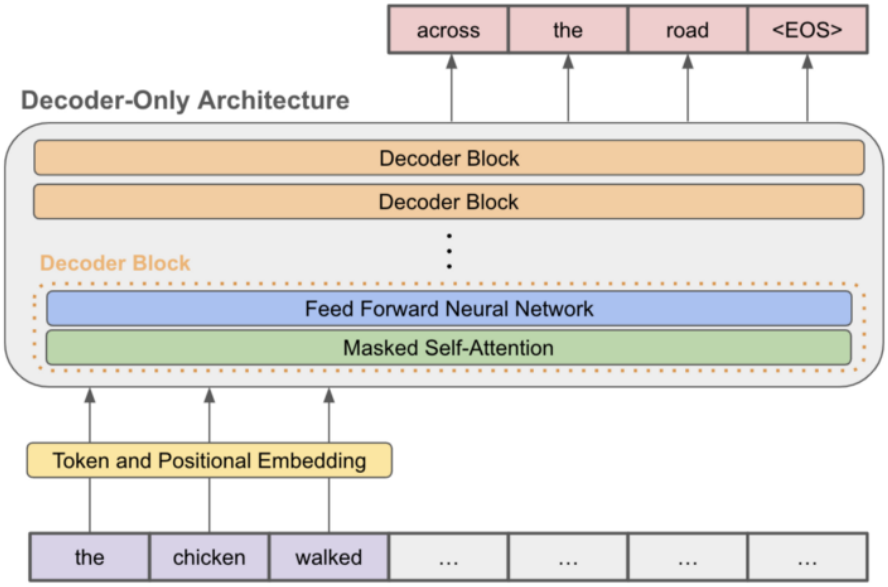

Figure 9: OPT Decoder Architecture [16]

Tokenizer: We used standard GPT2 pretrained tokenizer (medium) with the maximum length capped at 64. OPT model supports the GPT2 pretrained tokenizer with slight modification in few tokens.
Optimizer: We used the Adam optimization scheme with a max learning rate of 1e-4.
Scheduler: We used Linear Scheduler with warmup to increase the learning from a small learning rate to a constant with a warmup step of 10000.

We trained it for 32000 iterations, the training loss was 10.62. It was performing worse than GPT2. Following are some examples of limericks generated by the model.

im a and to is the@ as

and an
an in that a it and@ a you
in of a i
a and@ you@
as is my with @
and a@@ a
isthat you
a
its all that the@ and

Accuracies

We experimented with several setups,
1) Generating limericks with GPT-2 and Cross Entropy loss
2) Generating limericks with OPT and Cross Entropy Loss
3) Generating limericks with GPT-2 and Rhyme score + Context score for filtering
4) Generating limericks with GPT-2 and Rhyme score + Context score for filtering + RhymeBERT for fixing

This is our final loss curve followed by some generated samples from our final setup:

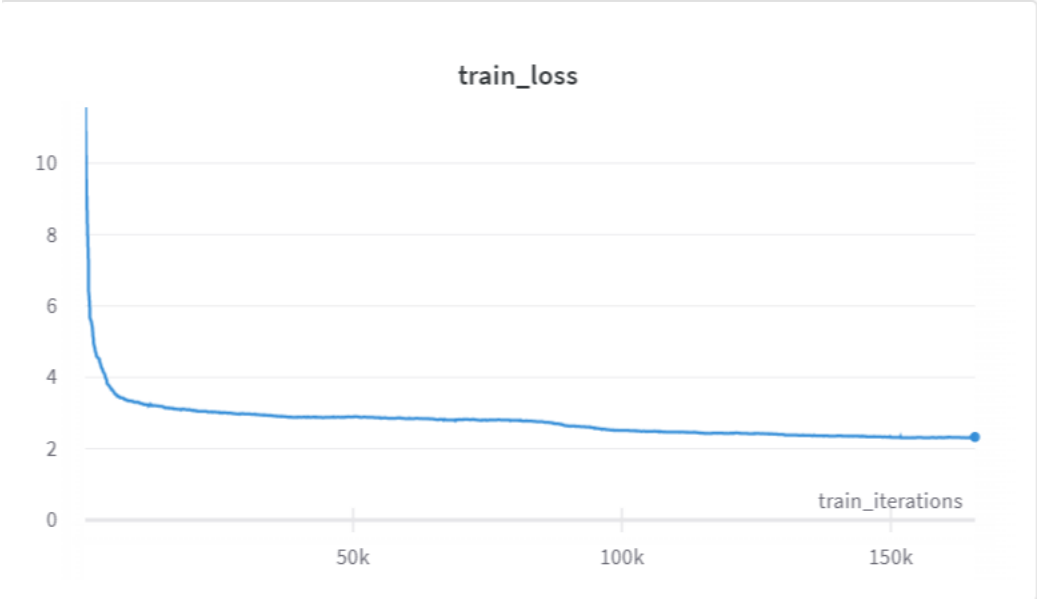

Figure 10: GPT-2 Loss Curve (160,000 iterations, 145,000 training Limericks

Some folks get a kick out of fooling,
with jokes that fool people theyre doing
of course they are sweet,
since the people they meet
with their jokes are not usually amusing

Some folks get a kick out of fooling,
with jokes that fool people theyre doing
of course they are sweet,
since the people they meet
with their jokes are not usually amusing

Here's my plan lets combine our forces,
with a force thats both mighty and courseless
I'm as sharp as a tack,

and I won't leave our slack
I'm your captain your marshal your forceless

4.2 RhymeScorer

RhymeScorer is a Siamese LSTM based classifier that has been trained on approximately 150,000 pairs of rhyming and non-rhyming words. The training data was created by using the limericks datasets - Sam Ballas Dataset[7] and Zenodo Dataset of limericks for computational poetics[12]. The limericks from these datasets were combined and the last words from each of lines in the limericks were extracted. Finally, based on AABBA scheme, 4 rhyming and 6 non-rhyming pairs were extracted from each limerick.These 10 word pairs from each limerick were given labels 1 and -1 for rhyming and non-rhyming respectively. Before passing the word pairs to the Siamese LSTM classifier, they were converted into Phonemes using DeepPhonemizer model[11].

The model was trained for 7 epochs using AdamW optimizer and it attained an accuracy of 97% on the validation dataset. The validation loss declined from 0.15 in the first epoch to 0.013 in the 7th epoch. The figure below illustrates the loss dropping by epochs.

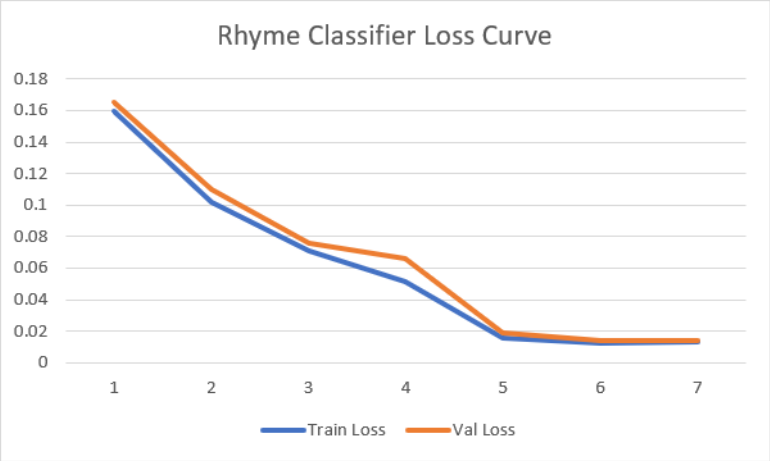

Figure 11: Rhyme Classifier Loss Curve

4.3 ContextScorer

For the Context Scorer, we used a pre-trained sentence transformer called "all-MiniLM-L6-v2". For all five lines, sentence embeddings of size 384 are generated using this sentence transformer. After that, pairwise cosine similarity between these embeddings is computed. Cosine similarity is computed as below:

$$\cos(\mathbf{t}, \mathbf{e}) = \frac{\mathbf{te}}{\|\mathbf{t}\|\|\mathbf{e}\|} = \frac{\sum_{i=1}^{n} \mathbf{t}_i \mathbf{e}_i}{\sqrt{\sum_{i=1}^{n} (\mathbf{t}_i)^2} \sqrt{\sum_{i=1}^{n} (\mathbf{e}_i)^2}} \tag{1}$$

To quantify the context, we calculate the add the cosine similarity between the first four lines paired with the fifth line. The range of context scores can be anywhere between 0 to 4.

4.4 Limerick Score Evaluation

Below is an example of how the limerick is scored.

Limerick Scoring Examples

Limerick	Rhyme Score (0-4)	Non-rhyme Score (0-6)	Context Score (0-4)
Some folks get a kick out of fooling With jokes that fool people they're doing Of course, they are sweet Since the people they meet With their jokes are not usually amusing	4	6	1.58
An old witch made her mark on the street Her face was the symbol of heat But the touch made her frown Saying oh dear oh the clown She was cursed and had witchlike feet	4	6	1.51
For a change I can just get my hand A huge shiny watch its a band When the power goes low A lights out then I know When the hours all run out of my hand	4	6	1.37
A choruss a genus of small Flowers with petals of pink and gold You might see one next door That is blooming this year If you travel through dallas or nome	0	6	0.33
When an acetol of somerene was given A chemical change could be laid bare With acetols we choose A high c with hs This was alizene yes its not afraid	0	6	0.32

Figure 12: Overall Limerick Score

4.5 RhymeBERT

We fine-tuned the BERT model on MLM task by masking the last words of 150,000 limericks in the training dataset. We tried 3 ablations with this model. The losses and descriptions are documented in the figure below.

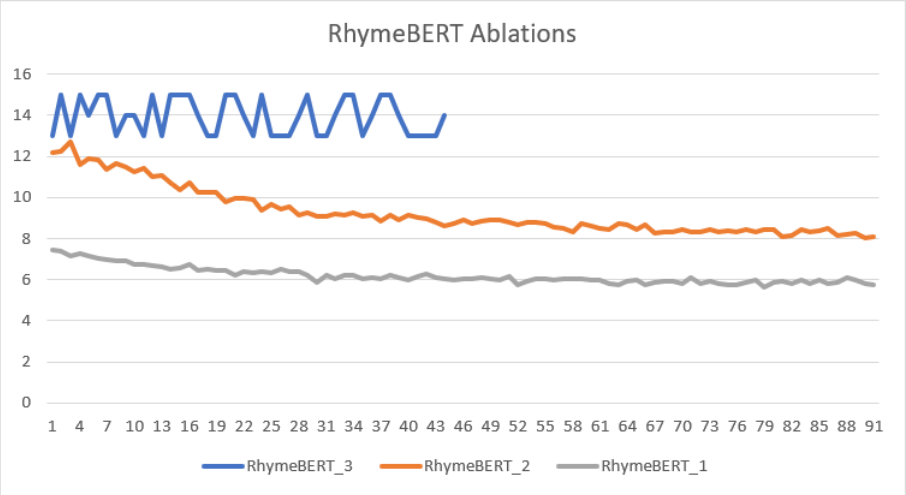

Figure 13: RhymeBERT Ablations

RhymeBERT 1 uses AdamW Optimizer with Linear with warmup scheduler and 2e-5 initial learning rate. This variant performed the best. In contrast, RhymeBERT 2 was trained with Adam optimizer and StepLR scheduler, it was not as effective in reducing the loss. RhymeBERT 3 performed the worst. It had no scheduler and a high initial learning rate of 1e-2.

5 Conclusion

We introduced a transformers based pipeline for limerick generation, evaluation and correction. The idea was to utilize deep learning models to generate limericks that appeal to humans. While our approach sets up a good baseline for limerick generation and evaluation, there is scope to further

improve and build on it. Firstly, our current pipeline focuses on context and rhyming. However, there are various other aspects of limericks that can be included like figure of speech, punchline, etc. Secondly, we can improve on our correction model to further enhance its ability to improve a limerick. Lastly, the current pipeline is generation-filtration. There are possibilities to somehow teach the model to rhyme training by penalizing it for not adhering to rhyme scheme. The possibilities with art generation using NLP are endless and hold immense business and real-world applications including but not limited to customized marketing, education, etc. With our work, we were able to generate decent quality limericks and successfully quantify two important aspects of rhyming and context.

Acknowledgement

We want to thank Dr. Rita Singh and Dr. Bhiksha Raj (Carnegie Mellon University) for their continuous guidance and support. Additionally, we also want to thanks Yashash Gaurav for his ongoing mentor-ship on our project.

References

[1] Alexander, J.A. & Mozer, M.C. (1995) Template-based algorithms for connectionist rule extraction. In G. Tesauro, D.S. Touretzky and T.K. Leen (eds.), *Advances in Neural Information Processing Systems 7*, pp. 609–616. Cambridge, MA: MIT Press.

[2] Bower, J.M. & Beeman, D. (1995) *The Book of GENESIS: Exploring Realistic Neural Models with the GEneral NEural SImulation System.* New York: TELOS/Springer–Verlag.

[3] Hasselmo, M.E., Schnell, E. & Barkai, E. (1995) Dynamics of learning and recall at excitatory recurrent synapses and cholinergic modulation in rat hippocampal region CA3. *Journal of Neuroscience* **15**(7):5249-5262.

[4] Previous PoetAI work : https://github.com/YashashGaurav/poetai-public

[5]T. Nguyen, P. Nguyen, H. Pham, T. Bui, T. Nguyen and D. Luong, "SP-GPT2: Semantics Improvement in Vietnamese Poetry Generation," 2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA), 2021, pp. 1576-1581, doi: 10.1109/ICMLA52953.2021.00252.

[6] Kai-Ling Lo, Rami Ariss and Philipp Kurz (2022), "GPoeT-2: A GPT-2 Based Poem Generator", `https://arxiv.org/pdf/2205.08847.pdf`

[7] S. Ballas, "PoetRNN," Available at https://github.com/sballas8/PoetRNN/ (2015)

[8] Jianyou Wang, Xiaoxuan Zhang, Yuren Zhou, Christopher Suh, Cynthia Rudin There Once Was a Really Bad Poet, It Was Automated but You Didn't Know It. Transactions of the Association for Computational Linguistics 2021; 9 605–620. doi: `https://doi.org/10.1162/tacl{_}a{_}00387`

[9] N.A., `https://developers.google.com/machine-learning/gan/discriminator`

[10] L.S. Marvin (2018) Siamese LSTM, `https://github.com/MarvinLSJ/LSTM-siamese`

[11] cschaefer26 (2021), DeepPhonemizer, `https://github.com/as-ideas/DeepPhonemizer`

[12] Zenodo Dataset , `https://zenodo.org/record/5722527#.Y0szkOzMK3I4`

[13] Prosodic, a metrical-phonological parser written in Python, `https://github.com/quadrismegistus/prosodic`

[14]Understanding the open pre-trained transformers (OPT) library. (n.d.). Retrieved November 15, 2022, from `https://towardsdatascience.com/understanding-the-open-pre-trained-transformers-opt-library-193a29c14a15`

[15]Decoder-only architecture used by GPT-2. - researchgate. (n.d.). Retrieved November 15, 2022, from `https://www.researchgate.net/figure/Decoder-Only-Architecture-used-by-GPT-2_fig1_349521999`

[16]The annotated GPT-2. Committed towards better future. (2020, February 18). Retrieved November 14, 2022, from `https://amaarora.github.io/2020/02/18/annotatedGPT2.html`

[17] Jey Han Lau et al. "Deep-speare: A Joint Neural Model of Poetic Language, Meter and Rhyme". In: (2018). DOI: 10.48550/ARXIV.1807.03491. URL: https://arxiv.org/abs/1807. 03491.

[18] all-MiniLM-L6-V2, `https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2`

[19]ApurbaSengupta. (n.d.). Apurbasengupta/Lyrics-generation-using-bert: Generating English rock lyrics using bert. GitHub. Retrieved December 9, 2022, from `https://github.com/ApurbaSengupta/Lyrics-Generation-using-BERT`

[20]Bert 101 - state of the art NLP model explained. BERT 101 - State Of The Art NLP Model Explained. (n.d.). Retrieved December 9, 2022, from https://huggingface.co/blog/bert-101

[21]Oedilf. (n.d.). Retrieved December 14, 2022, from http://www.oedilf.com/db/Lim.php

[22]Abdibayev, A., Riddell, A., Igarashi, Y., amp; Rockmore, D. (2021, September 21). Dataset of limericks for computational poetics. Zenodo. Retrieved December 14, 2022, from https://zenodo.org/record/5722527.Y0szkOzMK3I

[23]Horev, R. (2018, November 17). Bert explained: State of the art language model for NLP. Medium. Retrieved December 14, 2022, from https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270