# Generating Well-Formed Limericks and Haikus with GPT-2

**Bardia Shahrestani**
bardia.shahrestani
@mail.mcgill.ca

**Matthew Korahais**
matthew.korahais
@mail.mcgill.ca

**Daniel Korsunsky**
daniel.korsunsky
@mail.mcgill.ca

## Abstract

We explore the capabilities and limits of GPT-2 in the case of well-formed poems, specifically limericks and haikus. We hypothesized that GPT-2 trained without phonetic annotations would be unable to systematically learn and generate syllabic patterns and rhyme scheme, since these features are grounded in real world acoustic representations. Our model trained with list-of-rhymes annotations outperformed baselines, generating perfect-scoring limericks 33% of the time. Our best haiku model generated valid haikus in 29% of cases, with an average syllable error rate of <0.4. Our work invites further research into methods of combining text and phonetic data for more convincing text generation.

## 1 Introduction

There has been significant success in state-of-the-art text generation using large language models like GPT-3 (5) and BERT (7), including for specific genres of text. We wanted to explore the capabilities of large language models when it comes to structured poetry. Poetry is an interesting language limit case, in which our conceptions (models) of phonetics, syntax, semantics, and pragmatics are most tested.

Since free-verse poetry is difficult to quantify computationally, we chose to focus on generating structured limericks and haikus as they have well-defined rules. Limericks have 5 lines and follow an AABBA rhyme scheme with about 7-9 syllables in lines 1, 2, and 5, and 5-7 syllables in lines 3 and 4. Haikus have three lines and a 5-7-5 syllable count. Strictly speaking, limericks are *anapaestic*, meaning that the longer lines have three "beats" and the shorter lines have two "beats", but this was outside the scope of our exploration and thus approximated by syllable counts.

Accordingly, our research question was:

*To what extent can GPT-2 learn and generate well-formed, valid poetry, which depends on phonetic features like syllabic structure and rhyme scheme?*

We chose to use GPT-2 as it is widely available and easy to finetune. Since GPT-2 only uses text data, we suspected that models trained on limericks and haikus without phonetic annotations would be unable to systematically learn and generate syllabic patterns and rhyme scheme: especially in the English language, there is often considerable mismatch between spelling and pronunciation. Indeed, recognition of rhythm and rhyme is *grounded* in acoustic representations in the real world and thus extremely difficult, if not impossible, to infer from text data alone (8).

For limericks, we trained three models: a non-annotated baseline model, and two models trained with phonetic and list-of-rhymes annotations, respectively. Judged on the limerick criteria, the rhymitized model significantly outperformed the other two, producing perfect-scoring limericks 33% of the time (vs. 0% and 1.4%, respectively). For haikus, we evaluated one model trained on non-annotated data at 1000, 2000, and 3000 steps and found that the 3000-step model outperformed the other two, generating valid haikus 29% of the time and having the lowest syllable error rate.

These results confirmed our hypothesis that models trained on limerick and haiku text data without phonetic information would be unable to learn and generate syllabic patterns and rhyme scheme, while those with phonetic annotations would be more successful.

## 2 Related work

### 2.1 GPT family

In recent years, OpenAI's GPT-2, and its recent successor GPT-3, have blown the NLP state-of-the-art

language models out of the water with their ability to generate human-like text (4) (5), having been trained on 1.5 billion and 175 billion parameters, respectively. Despite the incredible potential of these models, the fact that they are trained solely on text means they still face a grounding problem (8), which in the case of poetry prevents understanding of phonetic-based features, such as rhyme and syllable structure, as well as semantic and pragmatic features.

### 2.2 Poetry with GPT-2

There have been many successful attempts at generating poetry with GPT-2 (10) (9), to the extent that some have been turned into books (11). However, attempts at introducing rhyming schemes have either been done by creating custom tokenizations for the underlying transformer model and enforcing the model to choose among those (12) (13) or they enforce rhyming by directly filtering the verses for the valid rhyme where furthermore they do not generate all verses of a poem (3). In this regard, we may be the first to solve rhyming using data augmentation alone.

## 3 Datasets

### 3.1 Limerick dataset

For our limerick model, we used the "Dataset of limericks for computational poetics" compiled by Abdibayev et al. (1). The dataset is comprised of 98k limericks scraped from The Omnificent English Dictionary In Limerick Form (OEDILF). It is filtered to only include poems that passed the basic requirements of limerick form, including having five lines and no emojis or symbols. According to the authors, each limerick was written by a human contributor and evaluated by "rigorous moderation". As far as we could tell, each limerick indeed passes all the requirements, including having 5 lines, an AABBA rhyme scheme, and appropriate syllable counts.

As a warning, as is often characteristic of limericks, the data contains some vulgarity that our models sometimes picked up on. While it might have been preferable to avoid these results, this was by far the most comprehensive and structured limerick data we could find.

### 3.2 Haiku dataset

Our haiku data was borrowed from Harshit Jhalani's Haiku Dataset on Kaggle (2). The dataset is comprised of over 144k three-line English poems, with one line per column in the csv file. Since many poems did not follow the 5-7-5 syllable structure of the standard haiku, we used NLTK's `cmudict` interface to count syllables and separate the data into 112k good haikus and 32k bad haikus. Only the good haikus were used to train the GPT-2 model. All of the poems were annotated with additional information, such as authorship, that was excluded from training.

Approximately 2/3 of the "good" haikus were generated by TWAIKU, a Twitter bot that creates haikus out of tweets exactly 17 syllables long (5+7+5). Another study (9) chose to reject the TWAIKU data from the same dataset because of concerns for its authenticity. They did this at the cost of keeping a strict 5-7-5 syllable criterion, focusing instead of having human-evaluated semantic metrics. We decided such metrics are too subjective for our purposes, and thus opted to keep the TWAIKU data, as our primary goal was to evaluate the models' success with the poetic form (syllable count) rather than its content. Moreover, there are few comprehensive haiku datasets available, and these haiku-d tweets were no less coherent than their organic counterparts.

## 4 Methods

### 4.1 The GPT-2 Model

All of our models were finetuned versions of the GPT-2 text-generating model (using either the 124M, 355M, or 774M versions), specifically `gpt-2-simple` v0.7.2 in Google Colaboratory. We used either the Nvidia P100 or K80 GPU for all tasks. Each limerick and haiku was annotated with `<|startoftext|>` and `<|endoftext|>` tokens to disambiguate them.

### 4.2 Limerick Training

**124M non-annotated (bare) model:** This model was trained on the full limerick dataset, which was used to finetune the 124M (smallest) GPT-2 model for 1000 steps with a default learning rate of `1e-4`. The limericks were not annotated in any way beyond the start and end tokens. These default and comparatively weak parameters were chosen to establish a baseline model for comparison, to evaluate which features of limerick structure, if any, GPT-2 could learn.

**774M phoneticized model:** This model was trained on the full limerick dataset, used to finetune

the 774M GPT-2 model. The model was trained for 4315 steps on an `1e-5` learning rate. Each word in the limerick data was augmented with the `cmudict` phoneticization scheme. For example, the limerick:

> A PIRATE WHO'S GOOD WITH HIS SWORD
> CAN VANQUISH A MUTINOUS HORDE,
> THEN INDUCE THE BATTALIONS
> THAT GUARD SPANISH GALLEONS
> TO SMILE AND SAY "WELCOME ABOARD!"

Was annotated as:

> A [AH0] PIRATE [P AY1 R AH0 T] WHO [HH UW1] 'S [EH1 S] GOOD [G UH1 D] WITH [W IH1 DH] HIS [HH IH1 Z] SWORD [S AO1 R D] CAN [K AE1 N] VANQUISH ...

We used this phoneticization technique to evaluate if the model could learn patterns of end rhymes with similar pronunciations. Each limerick had to be set to lowercase, and punctuation was not annotated, to search entries correctly with the `cmudict` library. Certain rare words or misspellings did not have entries. During text generation, all phonetic annotations were removed.

**774M rhymified model:** This model fine-tuned the 774M (large) GPT-2 model, trained for 2000 steps on a `1e-4` learning rate, 4000 steps on a `1e-5` LR, 800 steps on a `3e-6` LR, and finally 3200 steps on a `5e-7` LR (10,000 total). For each limerick, the (rhyming) endline word was annotated with a list comprised of its existing rhymes from the poem along with random rhymes from the `pronouncing` dictionary, for a total of up to five rhyming words. During text generation, all rhyming annotations were removed. For example, the limerick from above was annotated as:

> A PIRATE WHO'S GOOD WITH HIS SWORD ['UNDERSCORED', 'HORDE', 'BOARD', 'IGNORED', 'EXPLORED']
> CAN VANQUISH A MUTINOUS HORDE ['IMPLORED', 'UNEXPLORED', 'SWORD', 'LABOURED', 'PORED'],
> THEN INDUCE THE BATTALIONS ['SCALLIONS', 'GALLEONS', 'MEDALLIONS', 'ITALIANS']...

### 4.3 Limerick Evaluation

Each model was evaluated at 5 temperature levels (T = 0.1, 0.3, 0.5, 0.7, 1.0). At each temperature, the model generated 100 poems which were graded on a scale of 0 to 3, earning one point for meeting each of the following requirements: (i) having five lines, (ii) lines 1, 2, & 5 having more syllables than lines 3 & 4, and (iii) following the AABBA rhyme scheme. The score counts were then averaged across the 500 limericks.

Syllable counts were computed using Python's English language `pyphen` syllable dictionary. Rhyme scheme was graded using the `cmudict` phoneticization interface, by which the last phonemes of endline words had to match. Although this is looser criteria than typically expected for rhyming, a stricter grading scheme would have been too restrictive for our purposes. In particular, clearly rhyming words often had slightly different ending phonemes according to `cmudict`, and it would have been too task-intensive to equate all "near-phonemes". The `pronouncing` dictionary would have been too restrictive as well, since many rhyming words do not appear in each others' entries (e.g. "galleons" does not appear in the entry for "battalions"). Semantic coherence of the poems was not evaluated.

### 4.4 Haiku Training

**355M non-annotated model:** Our only haiku model was trained on the "good" haikus (see Section 3.2), used to fine-tune the 355M GPT-2 model and evaluated at 1000, 2000, and 3000 steps, with a LR of `1e-4` for the first 1000 and `1e-5` for the last 2000. The haikus were not annotated in any way beyond the start and end tokens. Since there was no rhyming to learn, there was no reason to augment the data in any other way.

### 4.5 Haiku Evaluation

100 poem samples were generated at each temperature (0.1, 0.3, 0.5, 0.7, 1.0) at each of the step sizes (1000, 2000, 3000), for a total of 1500 haikus. For each step size, we calculated the percentage of haikus that met the 5-7-5 syllable criterion, averaging over temperatures.

The error in haiku results was also calculated for each step size *per line* as the absolute value difference between the number of generated syllables in a certain line (1, 2, or 3) and the correct number. For example, if Line 2 of a generated poem contained 6 syllables, the error for Line 2 would be 1. The *average total error* was also calculated for each step size, which can be interpreted as the average absolute value difference between the number of syllables in all generated poems and total number of syllables in a proper haiku (i.e. 17). Semantic coherence of the poems was not evaluated.

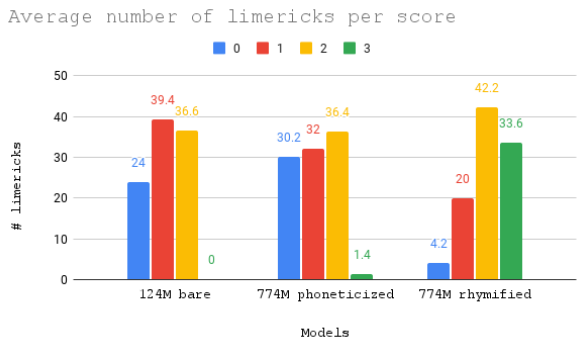


Figure 1: Percentage of limericks by score received

## 5 Results

### 5.1 Limerick Results

The models were evaluated by the scoring method outlined in Section 4.3. Results can be found in Figure 1. Details scores by temperature can be found in the linked source code. Almost all of the time, a score of 2 meant that the poem failed to rhyme (as opposed to failing one of the other metrics). A score of 1 most often meant that the poem was only successful in line count, since our evaluation for syllable count presumed that the poem had at least 5 lines (and most line count-failing poems had 4 lines or fewer).

**124M bare model:** As seen in Figure 1, the model was able to learn line counts and syllable counts/ratios fairly well with only 1000 steps and no annotation to the data. Scores for particular temperatures did not significantly deviate from the average. The model was successful in at least one of the metrics 76% of the time. However, none of the limericks rhymed successfully. For example:

> *The lightning attack dealt a blow*
> *To our city, where we were once proud.*
> *In the days of our great*
> *History, it seems,*
> *We were frequently robbed, none too proud.*

**774M phoneticized model:** This model was no better (worse, in fact) than the 124M bare model at learning and generating limericks with proper line counts and syllable ratios, despite training for over 4 times as long and at a lower learning rate. Scores for particular temperatures did not significantly deviate from the average. It only rhymed correctly 1.4% of the time (7/500 limericks) and was entirely unsuccessful 30.2% of the time.

**774M rhymified model:** This model significantly outperformed the baseline (bare) model and the phoneticization strategy, earning a perfect score 1/3 of the time. It also performed best in limericks scoring a 2 (no rhyming), as well as in avoiding scores of 0 (4.2%) or 1 (20%). Scores by temperature were more variable. Of the 100 limericks generated with T = 0.1, 63 received a perfect score, compared to only 6/100 generated with T = 1.0. Perfect-scoring limericks included:

> *The great actor's name is a curse,*
> *Since the roles that he's been given are worse.*
> *His films that were good*
> *Are now so poorly should*
> *That he's left in a state that's worse.*

However, since our rhyming criteria only compared the last phonemes of endline words (see Section 4.3), some perfect scoring limericks did not rhyme, such as *crook* and *take* in the following couplet (last phoneme 'K'):

> *A man who's a bit of a crook*
> *Has a way with women, you'll have to take*
> *...*

### 5.2 Haiku Results

The haiku models were evaluated as outlined in Section 4.5. As shown in Table 1, the model was able to generate between 6 and 39 valid haikus per 100 samples, depending on the parameters. The averages increased noticeably, the longer that the model was trained. Although only a minority of generated poems followed the syllabic structure of a haiku, it should be noted that the average generated poem differed from the valid syllabic structure by less than 1 syllable per poem, as outlined in Figure 2. More details concerning the outputs of the methods of haiku evaluation, including generated poetry samples and success and error calculations for each temperature can be found in Table 1 and in the appended data submitted with this assignment.

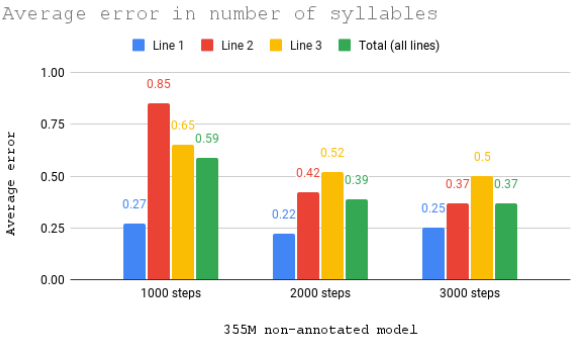


Figure 2: Average error in # of syllables in haikus

| No. steps | T = 0.1 | T = 0.3 | T = 0.5 | T = 0.7 | T = 1.0 | Avg |
|---|---|---|---|---|---|---|
| **1000** | 28 | 8 | 6 | 10 | 6 | 11.6 |
| **2000** | 14 | 35 | 26 | 29 | 13 | 23.4 |
| **3000** | 39 | 37 | 32 | 23 | 16 | 29.4 |

Table 1: Number of valid haikus by step count

An interesting trend, shown in the appended data, was that at T = 0.1, the Line 1 average error over all haikus at all three step sizes was 0. Scanning the sampled data, we found very repetitive output for poems at temperature 0.1, with every sampled poem beginning with the line "I'm so tired of" or "I'm so happy I".

## 6 Discussion and Conclusion

### 6.1 Limerick Discussion

Unsurprisingly, the 124M bare and 774M phoneticized models were reasonably successful in learning proper line count and syllable ratios between A lines and B lines, as GPT-2 has demonstrated significant success in picking up on these (near) syntax-level features. Because limericks have a loose syllable requirement, it was not extremely difficult for the model to use line length as a sufficient proxy for syllable counts.

The lack of rhyming success was also unsurprising, especially because we were working with English language data, in which there is often considerable mismatch between spelling and pronunciation. The fact that the 774M phoneticized model was so much larger and longer trained without improvements shows that GPT-2 would have no reason to pick up rhyming patterns unless phonetic information was encoded in the input text. The poor performance of the model was somewhat surprising, but it is possible that annotating *every* word with its phoneticization added unnecessary noise. Moreover, as mentioned, `cmudict`'s method assigned rhyming words slightly different ending phonemes, which could have made all the difference when trying to learn correlations.

The 774M rhymified model's comparative and absolute success demonstrates that data augmentation was necessary for learning rhyming, compared to model size. Crucially, however, what the model must have learned is a certain schema of *word association* rhyming rather than *phonetic* rhyming. Nevertheless, this is an exciting finding as it demonstrates a new way of connecting related words that makes up for the shortcomings of text-only data. As far as we are aware, we are the first to make GPT-2 rhyme by using data augmentation, as opposed to creating a custom list and forcing the model to choose from possible rhymes (see Section 2).

### 6.2 Haiku Discussion

The fact that generated poems were on average no more than one syllable away from being valid haikus is to be expected, given the text-based nature of the training data. This indicates that the model had little trouble picking up on the average length per line in the haiku training dataset. Indeed, even after only 1000 steps, the model was already producing output with nearly valid haiku form. The model showed considerable improvement with more training, showing that our model indeed learned the syllable count of haikus without being fed any phonetic information.

A possible explanation for the zero error at T = 0.1 was that the repeated phrases such as "I'm so tired of" were the most frequent ways that poems (tweets) from the TWAIKU dataset began, although further investigation is required.

### 6.3 Limitations and Further Exploration

Pursuing the success of the rhymified model, we may be able to make further improvements by regenerating rhyming tags with different samples from the rhyming dictionary every couple thousand steps to broaden the scope of acceptable rhymes.

More generally, further research could address the grounding problem by appending sound files to the poetry data. This technique could be more generalizable, make rhyming more robust, and even account for other features of poetry, such as rhythm, emphasis, meter, and mood.

## Statement of Contributions

**Bardia Shahrestani**: Training and testing limerick models, creating the limerick tester and annotation techniques, applied philosophy of language
**Matthew Korahais**: Training and testing haiku models, creating the haiku tester and preprocessing techniques, literature review
**Daniel Korsunsky**: Researching and collecting datasets, training baseline limerick model, contributing to optimization techniques, data visualization, report writing, project and group management

## References


[1] Abdibayev, Almas, Allen Riddell, Yohei Igarashi, and Daniel Rockmore. 2021. Dataset of limericks for computational poetics (Version v3). doi: 10.5281/zenodo.5722527

[2] Jhalani, Harshit. Haiku dataset. www.kaggle.com/hjhalani30/haiku-dataset

[3] Uthus, David Voitovich, Maria Mical, R.. 2021. Augmenting Poetry Composition with Verse by Verse.

[4] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I. 2019. Language Models are Unsupervised Multitask Learners.

[5] Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A. and Agarwal, S., 2020. Language models are few-shot learners. arXiv preprint arXiv:2005.14165.

[6] Nichols, Eric, Leo Gao, and Randy Gomez. "Collaborative Storytelling with Large-scale Neural Language Models." Motion, Interaction and Games. 2020. 1-10.

[7] Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).

[8] Bender and Koller. Climbing towards NLU: On Meaning, Form and Understanding in the Age of Data. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. https://www.aclweb.org/anthology/2020.acl-main.463/

[9] Miceli, Giacomo. "Haiku Generation A Transformer Based Approach With Lots Of Control." 2020. https://www.jamez.it/blog/wp-content/uploads/2021/05/Haiku-Generation-A-Transformer-Based-Approach-With-Lots-Of-Control.pdf

[10] Branwen, Gwern. GPT-2 Neural Network Poetry. 2019. https://www.gwern.net/GPT-2

[11] Hseih, K., 2019. Transformer Poetry: Classic Poetry Reimagined by Artificial Intelligence. [online] https://papergains.co/pdfs/Transformer-Poetry-978-1-7341647-0-1.pdf [Accessed 17 December 2021].

[12] Summerstay. True poetry. GitHub repository. https://github.com/summerstay/true_poetry. [Accessed 19 December 2021].

[13] Summerstay. Poem generator. GitHub repository. https://github.com/summerstay/true_poetry. [Accessed 19 December 2021].