Pushkin.AI: Generating Russian Poetry Using Deep Learning

Priyanshi Garg
Carnegie Mellon University
pgarg2@andrew.cmu.edu

Shravya Nandyala
Carnegie Mellon University
snandyal@andrew.cmu.edu

1 Introduction

What makes art human? Is it the fruition of idea, or its arduous completion? Perhaps it is the artist themselves, embedding their very humanity into the inanimate, bringing it to life. Regardless of the answer, it is undeniable that art is a human process that cannot live without the artist. So, we ask not if a machine can create art, but rather: to what extent can a machine become an artist? Can we use machines to fulfill human needs that are satisfied by artists?

Russian poetry is vastly revered for its vivid, thought-provoking verses and text. Specifically, the Golden Age of Russian poetry, characterized by poets like Pushkin, is still extensively studied and examined today. Consequently, to answer our pressing questions about art and machines, we attempt to create a deep-learning architecture that takes in a seed text and outputs a Russian poem. The training dataset consists of a corpus of filtered quatrains in the Russian Language to train the Generative Pre-Trained Transformer (GPT). To fine-tune our model to the style of a specific author, we filtered an existing corpus of 19,000 poems for a resulting 763 poems written by Alexander Sergeyevich Pushkin. Taking the seed text as a starting point, the model outputs a poem on the topic of the given seed text. The results were evaluated on relevance to the seed text, rhyme scheme adherence, grammatical coherence, meaning, as well as style and aesthetic. The goal of the model is to generate text which is poetic in both meaning and form.

2 Background

For our baseline, we utilized a stacked bi-directional LSTM (Long Short-Term Memory RNN) due to its promising results in Russian poetry generation found during the literature review. The model consisted of an embedding layer used to construct vectorized representations of each sentence in our corpus. This was followed by two LSTM layers with dropout to minimize overfitting, and a ReLU activation and softmax output layer. From our trained model, 5 poems were generated of 30 words each using next-word prediction. After 100 epochs, the highest training accuracy achieved was 74.5% with the lowest cross-entropy loss of 1.34. Though this produced some good initial results, to improve the quality of poems and truly differentiate our model from a regular text generation model, we switched to a transformed-based architecture and incorporated rhyme and phonetics into our model.

3 Related Work

Several pieces of literature as well as code helped guide our model and process:

1. **Transformer-Based Architecture**: There has been some work done in the past two years that suggests that using a transformer-based approach like the Generative Pre-Trained Transformer model (GPT) for the task of poetry generation yields good results [4]. Furthermore, the parallelization capability of a transformer allows us to use a larger amount of text and incorporate more data.
2. **Phonetics**: One aspect that makes Russian poetry unique from other languages is the importance of "stress" in words and how it impacts the rhyme scheme, structure, and

aesthetic of a poem. Inspired by previous research done on this in Russian poetry, we aimed to encode information about stress, semantics, and phonetics of a poem [5] [1].

3. **Codebase**: We utilized an existing pre-trained (GPT + T5) model that explicitly learns the phonetics of Russian speech like stressed and unstressed syllables [3] and was inspired by work done on Edgar Allen Poe stylized poetry generation to fine-tune our model based on Pushkin's work [2].

4 Methods

Our central motivation was to distinguish our generative model from a regular text generation model for which there has been extensive work done already. To do this, we decided to explicitly incorporate the phonetics of the Russian language into our model.

4.1 Stress Marking Tokenizer

In addition to containing grammatically correct content, the text needs to contain a certain specific form called **Syllabo-Tonic Versification** [8]. This form is a way of organizing a poem, in which stressed and unstressed syllables alternate in a certain order, unchanged for all lines of the poem which is fairly common in Russian poetry. In addition to this form, Russian poems also tend to contain a phonetic correspondence of the stressed endings of the last words in the lines [6]. This information cannot be learned adequately by just fine-tuning a ruGPT (Russian language GPT) model. Hence, we need a way to explicitly teach our model Russian phonetics.

In order to accomplish this goal, we need to choose the correct representation of the text in order to explicitly represent knowledge about stressed and unstressed vowels which can be done through tokenization. ruGPT represents text as a sequence of some tokens. The number of these tokens is selected in advance (> 50,000 usually), and the dictionary is formed automatically so that any word of the language is represented by a pre-fixed set of tokens. We break words explicitly into normal syllables and also mark the stressed syllables with some kind of marker. Thus, two syllables that are identical except for stress become different tokens and during training, the model will understand where which one is used.

4.2 Incorporating Rhyme

In autoregressive models, like GPT, the text is generative from left to right. For poetry, this means that the last word is chosen after the initial sequence has already been generated, and hence the possibility for rhyme is limited. To tackle this problem, the model first selects the last word in the line with the desired rhyme pattern and then generates the rest of the sequence.

4.3 Training the Syllabo-Tonic Language Model

- Pre-train: First, the model trains on a very large set of texts without manual markup. The corpus for the pre-train in our case consists of automatically tagged Russian-language prose with a volume of approximately 10 billion characters.
- Fine-tune: The corpus used for this consists of an automatically tagged set of verses with marked verses and filtering low-quality verses.

4.4 Poet Style

As a further addition, we attempted to use the syllable-tonic language model as a baseline to capture the essence of a specific poet. For our purposes, we used the prior work of Alexander Sergeyevich Pushkin to fine-tune our model before generating our poems. To do this, we used an existing corpus of 19,000 Russian poetry and filtered it to 763 Pushkin poems, and passed a subset of it through the pre-trained syllable-tonic model using the custom tokenizer mentioned above. The model seemed to capture some information about the Pushkin literature but we suspect that the format of the data we passed caused some inconsistencies with the stress marking tokenizer and we did not have enough time to debug this issue. Hence, for empirical expert evaluation, we utilized stanzas generated before fine-tuning the model on Pushkin's work.

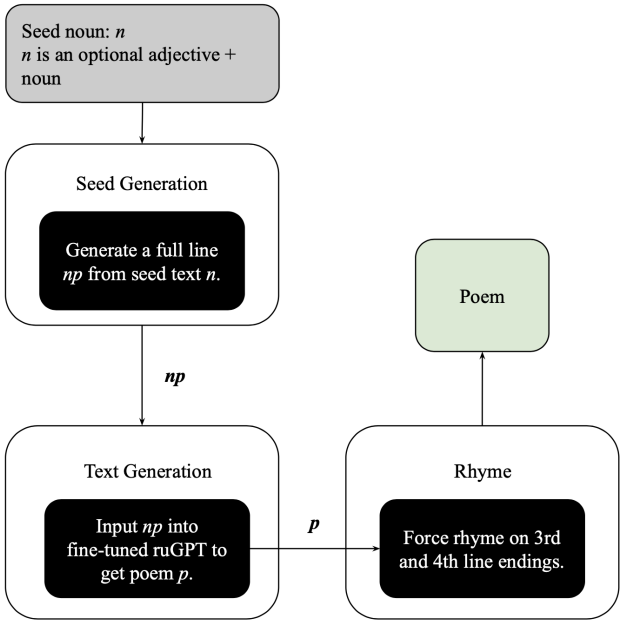

Figure 1: System Diagram

The exact process of the construction of a quatrain looks like this (also illustrated in Figure 1):

- Step 1: A fine-tuned version of ruGPT on keyword-sentence data is used to generate the first two lines of the quatrain.
- Step 2: The last words of the next two lines are picked based on the endings of the first two lines generated in step 1. Two words are considered rhyming if the stress falls on the same syllable, counted from the end, and also if the stressed and subsequent syllables are pronounced similarly. This check is done with an accentuator. It predicts the position of stress in arbitrary words using both a dictionary for frequency words and an ML model. The second check involves the transition from a symbolic representation to a phonetic one and is performed using a transcriptor.
- Step 3: After choosing the last words, we need to construct the rest of the remaining two lines. We want to do this in a way where the result is somehow relevant to the first two lines, and also syntactically combined with the selected keywords. We do this by training a special language module that works from right to left. This will build the line generation from the last word to the first. For this, the T5 model architecture is used. It combines a BERT-like bidirectional encoder and a GPT-like unidirectional decoder. One key feature of T5 is that it is trained to perform denoising or fill in deleted text fragments which closely aligns with our problem setting.

5 Results

We first selected 5000 lines from a corpus of Pushkin's poetry at random. Then, we fine-tuned our base model on this data for 6 epochs, with a learning rate of 0.0005 and batch size of 4. The training took an average of 68.73 minutes and the testing took an average of 3.01 minutes per epoch. The average loss across each epoch for the training/testing stages is shown in Figure 2.

Upon replacing our base model with the fine-tuned model, we encountered many issues. To fine-tune, we had to introduce some special tokens in order to parse the input data (eos token and pad token, for example). This allowed our fine-tuning code to process data and feed it through the model. However, because we did not have access to the pre-process used to fine-tune ruGPT to create the base model,

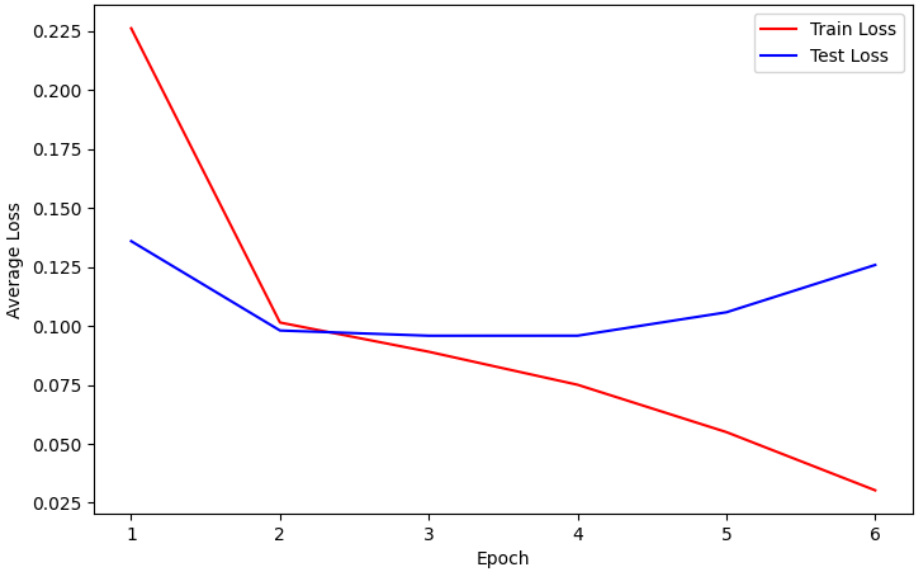

Figure 2: Loss Graph

we could not guarantee that the data we were using was in the correct format for the model to train on. This became an issue when we attempted to load the fine-tuned weights for the base model. Because of the added special tokens, the dimensions of the weights changed. To resolve this, we removed the extra tensors and resubmitted the weights to the base model but this time, the model only returned three-line poems instead of the usual four. We attempted to debug this but in the end, were unsuccessful in understanding how fine-tuning could cause such a big change. Part of our hypothesis as to why this issue occurred is that the base model itself is under development and contains bugs. It is possible that there is a related bug that is triggered when foreign weights are loaded into the base model. Because of this issue, the rhyme module was unable to force rhymes for the 3rd and 4th lines and the model ultimately failed to generate a poem.

Since we were unsuccessful in our attempt to fine-tune the model on Pushkin data, we instead to using the pre-trained model to generate our poems.

We gave our model five seed texts. The chosen seed texts are each part of titles from five Pushkin poems.

1. Title: Я вас любил – Seed text: любовь (tr: love)
2. Title: Ночь – Seed text: ночь (tr: night)
3. Title: Любопытный – Seed text: любопытство (tr: curiosity)
4. Title: Птичка – Seed text: птичка (tr: birdie)
5. Title: Зимнее утро – Seed text: зимнее утро (tr: winter morning)

The results of each generated text are included in Appendix A.

In comparing these results with the baseline results reported in our Midway Report, we find that the current model performs better on all measured factors. For grammar, the current model was found to be almost entirely grammatically correct whereas the baseline often failed to have a concrete subject / verb in a given line. For meaning, the current model is shown to stick closely to the seed text. However, the baseline used a bi-directional LSTM which was prone to shifting away from the initial focus because of the LSTM's poorer long-term dependency capturing. For rhyme, the current model enforces a consistent rhyme scheme whereas the baseline had no enforced rhyme. The current model also produces stanzas with more form (stanza is divided into four distinct lines), allowing a flexible word count for the text to be coherent. In the baseline model, the output was a standard word count

as opposed to line count which sometimes hindered creating a meaningful poem within the limit. We can conclude that the current model is a great improvement from the baseline.

6 Discussion and Analysis

6.1 Evaluation

After looking at several commonly used evaluation metrics for text-generation systems like BLEU, Perplexity, Rouge (Recall Oriented Understudy for Gisting Evaluation), LSA (Latent Semantic Analysis), etc, we concluded that none of them would be appropriate for our study since they do not effectively capture some of the key properties that we want our text to have which include originality, poetic form, and rhyme, grammatical correctness, and coherence. In line with other literature on Poetry Generation, we decided to use a human evaluation process. We included properties like qualities of poetry, closeness to seed text topic, and closeness to Pushkin's style [7]. To do so, we recruited three members of Carnegie Mellon University's Russian department faculty who are native/fluent speakers of the language.

Each evaluator was asked to score 10 four-line poems. Five of these were controls (excerpts from Pushkin's poetry from which the seed texts were chosen) and five were the generated poems (see Appendix A). The poems were presented in random order and for each, the evaluator was asked to score three factors that encompass "poetic" quality: grammar, meaning, and rhyme. Each of these was ranked on a scale of 1 to 5, 1 being non-grammatical / nonsensical / not rhyming and 5 being perfectly fluent/meaningful / perfectly rhyming. Then, the evaluator was asked how likely it was that Pushkin authored the poem, again, on a scale of 1 (not likely at all) to 5 (almost certain). Finally, the evaluator had to write a single word to describe the topic of the poem.

The findings of this evaluation are shown below.

Averages across Pushkin poetry.

	Evaluator A	Evaluator B	Evaluator C
Grammar	5	4.6	5
Meaning	5	5	5
Rhyme	5	4.6	5
Pushkin-ness	5	4	4.6

Averages across AI-generated poetry.

	Evaluator A	Evaluator B	Evaluator C
Grammar	4.8	4.2	4.75
Meaning	4.2	4.4	4.75
Rhyme	5	3.8	5
Pushkin-ness	1	1.8	1

Overall, the generated poems tended to be grammatical, meaningful, and rhyming. However, the Pushkin stylizing was unsuccessful and each evaluator was able to spot the different between AI-generated poetry and Pushkin poetry. One factor to consider is that a Russian academic will have studied Pushkin (perhaps extensively). As a result, their chances of recognizing a Pushkin poem based off of the text instead of style are perhaps higher than an average Russian speaker. On the contrary, a Russian academic may not be too familiar with Pushkin and thus, may not be able to accurately identify when his style conventions are being followed. Both of these reasons could explain the low scores for Pushkin-ness, along with a failure in the model.

When analyzing the words chosen by the evaluators to describe each poem, most were relevant to the seed text. The "birdie" seed text yielded answers "singer" and "talent" which reference the bird's song. The seed text "night" yielded answers "night" and "nocturne", and the seed text "winter morning" yielded answers "winter" and "thaw". Thus, we can conclude that the seed text was, for the most part, appropriately incorporated into the poem's theme but perhaps distorted at times without more context.

6.2 Possible Model Improvements

1. Going beyond rhyme: Right now, our model mainly considers rhyme as the differentiating poetic factor. Grammar and meaning are also expected of prose, but it is simplistic to impose a standard rhyme scheme and say that turns prose into poetry. Rather, we can understand other factors that make poetry inherently poetic. For instance, taking into account rhythm and syllabic count structures could improve the form of our model's generated output.
2. Effectively capturing a poet's style: With the observed results, our model does not imitate a poet's style well, if at all. A solution to to improve the specific style of the poetry would be to fine-tune the model on a much larger dataset. Perhaps, we can implement a classifier that, once the main model has generated output, is used to verify that certain style requirements for a given poet have been met. This would help test our model's efficacy and allow us to iterate before moving on to user testing which is harder to control.
3. Architecture modifications: The current model only generates four or eight line stanzas. To better this, we could add a new input to allow the user to chose the length or style of poem. If the user chooses, for instance, haiku, the outputted verse should adhere to the conventions of haiku. Or, the user should be able to request a 10 line poem. Additionally, for the current task, the architecture could be modified to include a rhythm module which understands the tempo of each line. Making this cohesive throughout the stanza could improve the perceived "poetic-ness" of the final output.

References

[1] Yi-Chen Chen et al. "Phonetic-and-semantic embedding of spoken words with applications in spoken content retrieval". In: *2018 IEEE Spoken Language Technology Workshop (SLT)*. IEEE. 2018, pp. 941–948.

[2] Scott Duda. *Generating an Edgar Allan Poe-Styled Poem Using GPT-2*. 2022. URL: https://scottmduda.medium.com/generating-an-edgar-allen-poe-styled-poem-using-gpt-2-289801ded82c (visited on 10/31/2020).

[3] Koziev. *Generating Russian Poetry Using Large Language Models*. 2022. URL: https://github.com/Koziev/verslibre (visited on 11/21/2022).

[4] Kai-Ling Lo, Rami Ariss, and Philipp Kurz. "GPoeT-2: A GPT-2 Based Poem Generator". In: *arXiv preprint arXiv:2205.08847* (2022).

[5] Alexey Tikhonov and Ivan P Yamshchikov. "Guess who? Multilingual approach for the automated generation of author-stylized poetry". In: *2018 IEEE Spoken Language Technology Workshop (SLT)*. IEEE. 2018, pp. 787–794.

[6] Wikipedia. *Clauses*. 2022. URL: https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%83%D0%B7%D1%83%D0%BB%D0%B0 (visited on 12/07/2022).

[7] Xingxing Zhang and Mirella Lapata. "Chinese Poetry Generation with Recurrent Neural Networks". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 670–680. DOI: 10.3115/v1/D14-1074. URL: https://aclanthology.org/D14-1074.

[8] Viktor Maksimovic Žirmunskij. "II. SYLLABO-TONIC VERSIFICATION". In: *The theory of verse*. Ed. by E. Stankiewicz and W. N. Vickery. Berlin, Boston: De Gruyter Mouton, 1966, pp. 34–87. ISBN: 9783111398594. DOI: doi:10.1515/9783111398594-004. URL: https://doi.org/10.1515/9783111398594-004.

A Appendix

1. Seed text: любовь (tr. love)

   Опять сливаемся с тобою воедино.
   Для нас двоих любовь и счастье, ангел мой.
   Покрепче обними, я так хочу, любимый,
   Чтобы последний раз ты счастлив был со мной.

2. Seed text: ночь (tr. night)

   Почти стемнело, Мишка спит,
   Шагов не слышно во дворе,
   Луна пред зеркалом стоит,
   Чтоб приглядеться лишь ко мне.

3. Seed text: любопытство (tr. curiosity)

   Вы не боитесь тех, кто щиплет нос?
   Вот я сейчас сижу за барной стойкой,
   Из любопытства Вам задам вопрос:
   Хитрющих, жадных, страшных стало сколько?

4. Seed text: птичка (tr. birdie)

   И сказали мне друзья,
   Соловей мне петь нельзя.
   Я не понял, почему же,
   Я не слышал песен хуже?

5. Seed text: зимнее утро (tr. winter morning)

   Ещё февраль - большая птица,
   Ещё без туч, без облаков.
   Но начинает торопиться,
   Срывая ледяной покров.