Generation of Russian Poetry of Different Genres and Styles Using Neural Networks with Character-Level Tokenization

Ilya Koziev
SaluteDevices
inkoziev@gmail.com

Alena Fenogenova
SaluteDevices
alenush93@gmail.com

Abstract

Automatic poetry generation is an immensely complex task, even for the most advanced Large Language Models (LLMs) that requires a profound understanding of intelligence, world and linguistic knowledge, and a touch of creativity. This paper investigates the use of LLMs in generating Russian syllabo-tonic poetry of various genres and styles. The study explores a character-level tokenization architectures and demonstrates how a language model can be pretrained and finetuned to generate poetry requiring knowledge of a language's phonetics. Additionally, the paper assesses the quality of the generated poetry and the effectiveness of the approach in producing different genres and styles. The study's main contribution is the introduction of two end-to-end architectures for syllabo-tonic Russian poetry: pretrained models, a comparative analysis of the approaches, and poetry evaluation metrics.

1 Introduction

Automatic poetry generation is a challenging task that requires systems capable of handling multiple levels of language understanding, including deep comprehension of text, linguistic and world knowledge, common sense, creativity, and an awareness of syllabic and rhythmic structures.

As a form of artistic expression, poetry has been produced in numerous languages, each with its own unique poetic traditions and forms. While most poetry generation systems focus on English and Chinese, there are also efforts targeting other languages (Hämäläinen and Alnajjar, 2019; Hämäläinen et al., 2022; Chudoba and Rosa, 2024). However, the task of automatically generating poetry in Russian remains underexplored and presents unique challenges.

To address this gap, we explore neural network architectures for the automatic generation of syllabo-tonic*[1] Russian poetry. Specifically, we investigate whether transformer-based models can effectively handle end-to-end generation of Russian syllabo-tonic poetry across various genres, styles, and forms. Our analysis reveals that mainstream byte pair encoding (BPE) tokenization often fails to align well with the structural units of Russian syllabo-tonic versification. To address this, we propose and evaluate language models with character- and syllable-level tokenization, training and testing their performance on the poetry generation task.

We also conduct a detailed study of poetry metrics (subsection 5.1) and share our experiences using existing methods to assess the quality of generated poems. These methods include automatic evaluation (Table 3) of fluency and poeticness for several models, as well as human evaluation of the overall quality of poetry generated by models with character-, syllable-, and BPE-based tokenizations (Table 1).

The contributions of our work are as follows:

- We propose several architectures utilizing character-level tokenization, including the CharLLaMa model, based on the Llama architecture (Touvron et al., 2023), and the CharMamba model, based on the Mamba selective state space architecture (Kheradmand et al., 2023). We have released the weights for the CharLLaMa-1.3B [2] and CharLLaMa-2.6B [3] models;
- We compare character- and syllable-level language models with baseline language models after supervised finetuning (subsection 3.2) on diverse poetry genres;

[1] All poetry terms marked with * are defined in the Glossary in Appendix A.

[2] https://huggingface.co/ai-forever/charllama-1.3B

[3] https://huggingface.co/ai-forever/charllama-2.6B

- We developed and open-sourced a library for Russian poetry stress placement and meter evaluation.[4]

- We demonstrate that small-sized language models with syllable-level tokenization can compete with larger general-purpose models in poetry generation tasks.

2 Related work

Creativity has been shown to be closely linked to human intelligence (Frith et al., 2021), making computational creativity a compelling area of research (Colton and Wiggins, 2012), including the study of creativity in LLMs (Franceschelli and Musolesi, 2024). Generative poetry, related to artistic creativity (Ismayilzada et al., 2024), differs from other natural language generation domains (Gatt and Krahmer, 2018) by its special lexical and phonological constraints, as well as specialized metrics to evaluating the quality of generated poems (see Chen et al. (2024) as an example). Recent advancements in LLMs have significantly improved the quality of poetry generation, to the extent that humans often cannot reliably distinguish between poems authored by humans and those generated by LLMs (Porter and Machery, 2024).

Tokenization approaches. Despite the progress of current generative models, there remains potential for further improvement in the quality of poetry generation. One area of research is alternative tokenization methods for LMs that circumvent the shortcomings of the currently mainstream BPE tokenization. In the case of syllabic or syllabo-tonic poetry, improvements can be achieved by using character- or syllable-level tokenization (Belouadi and Eger, 2022; Yu et al., 2024; Chen et al., 2024).

Character- and byte-level tokenization has been used in various systems for automatic poetry generation based on recurrent neural networks (Zhang and Lapata, 2014; Yan, 2016; Xie et al., 2017; Hopkins and Kiela, 2017; Tikhonov and Yamshchikov, 2018). After the invention of the transformer architecture, its applicability with character-based text representation for poetry generation was also investigated (Belouadi and Eger, 2022). The need to train the transformer language model from scratch limits the availability of such experiments. In the case of English language, there are open-source foundation models pretrained on vast corpora: CANINE (Clark et al., 2022) and ByT5 (Xue et al., 2022). CANINE is a family of encoder transformer models with tokens corresponding to Unicode codepoints. This model was utilized by Zhang et al. (2024) in melody-to-lyrics generation system. ByT5 implements an encoder-decoder architecture with byte-level tokenization. An example of its use for generating Czech poetry is available in Chudoba and Rosa (2024).

Syllable-level tokenization is a specialized variant of subword unit tokenization. Its effectiveness for generating poetry has been studied for several languages: Italian (Zugarini et al., 2019), Czech (Chudoba and Rosa, 2024), Vietnamese (Nguyen et al., 2021). Similar to character-level tokenization, syllable-level tokenization necessitates either resource-intensive pretraining of a language model from scratch or additional finetuning of a pretrained model with byte-pair encoding tokenization.

Generative poetry evaluation. A comprehensive evaluation of generative poetry models, like other creative models for open-ended tasks, poses significant challenges. Metrics designed for reference-based tasks, such as machine translation, are often unsuitable for this purpose. While perplexity is a commonly used metric for assessing generative poetry models (Yan, 2016; Che et al., 2017; Zugarini et al., 2019; Zhang et al., 2023; Hu et al., 2024), it has notable limitations (Kuribayashi et al., 2021; Wang et al., 2022). A standard alternative is to evaluate and compare generated poems using human assessors, either experts or non-professionals. However, this approach is costly and difficult to scale. In this context, the LLM-as-a-judge method, which has been applied to evaluate poems (Zhang et al., 2024) and prose (Yang et al., 2024), offers a promising solution for creative computation tasks.

Poetic texts exhibit structural properties that are well-suited for formal evaluation, such as adherence to syllable count per line, regularity in the alternation of stressed and unstressed syllables (poetic meter*), and rhyme schemes*. A significant advantage of this approach is the potential for full automation. Corresponding metrics can be computed during the evaluation phase, as demonstrated by Nguyen et al. (2021); Possi et al. (2023); Chudoba and Rosa (2024).

[4] https://github.com/Koziev/RussianPoetryScansionTool

3 Data

To train and evaluate poetry generation models, we required a substantial amount of Russian poetry data. However, publicly available datasets, such as Shavrina and Shapovalova (2017); Plecháč et al. (2023), are limited in size and insufficient for training generative models, particularly those based on transformer architectures.

To address this limitation, we collected a large volume of amateur poetry from various Internet sources (Appendix E). These sources often lack editorial oversight, leading to frequent spelling and punctuation errors that can negatively impact model performance. To mitigate this issue, we developed a rule-based spelling correction algorithm to address the most common errors. Further details about this algorithm are provided in Appendix B.

The collected poems also frequently exhibit defects in adhering to poetic meter* and rhyme*. Since these defects cannot be automatically corrected, we excluded such samples from the finetuning dataset. To identify meter- and rhyme-related defects, we used our custom library, described in 3.3.

3.1 Pretraining Dataset

Our pretraining dataset consists of two parts: 1) prose texts and 2) poetry texts. All texts have been annotated for stress with the library described in 3.3. The sources of the prose samples are presented in Appendix E.

To ensure that various data types are well-represented in the pretraining texts, the poetic data was upsampled (He and Garcia, 2009), as it constituted only half the volume of prose data. Based on our experiments, a fourfold upsampling of poetry is near optimal: more aggressive upsampling leads to a significant increase in plagiarism in the generated text, as models begin to reproduce memorized training data.

The resulting dataset contains 65 billion characters. The prose and poetry texts were randomly mixed and segmented into 1024-character blocks, starting with either `<prose>` or `<poetry>` tokens to identify the content. This setup allows models to generate poetry without extra finetuning by simply using the `<poetry>` token and an optional seed fragment. However, to better control the poem's theme, style, and sentiment, instructive finetuning is needed.

3.2 Finetuning Dataset

Instructive prompts. All samples in the finetuning dataset consist of an instructional prompt with specific parameters paired with a poem. This approach enables flexible control over the generation process by allowing users to specify all requirements directly in the prompt. This distinguishes it from models that rely on keyword-based seeds, as commonly used in systems like Boggia et al. (2022).

To streamline the creation of instructional prompts, we automated the generation process using an LLM, leveraging the collected poems as input. Manual prompt collection is both time-expensive and resource-demanding, making our automatic approach more efficient. The LLM analyzes a given poem — examining its genre, structure, and other key elements before generating a synthetic prompt. The input provided to the LLM follows the following structure (example is translated from Russian):

> Analyze the poem below in the genre "GENRE". Identify the main character, central idea, author's message, key conflict, emotions, vivid metaphors, and all proper names in the poem. Insert these into the "TEMPLATE" to create a task for a poet. Output only the resulting task sentence: "POEM"

`TEMPLATE` refers to a syntactic variation incorporating elements such as sentiment, emotion, length, and poetic meter* to create diverse prompts. Additional examples, including the original Russian version, are provided in Appendix F.

Quality. The quality of the finetuning dataset significantly affects poem generation results. Consequently, we focused extensively on cleaning the collected data. The dataset preparation code contains procedures for correcting typographic defects, including normalizing spaces, correcting commas, spell checking (Appendix B), and a set of filters for rejecting obviously bad poems. The filters include a set of heuristics for detecting the most common defects such as the repetition of some particles, as well as checking for compliance with a number of poetic rules. The latter is implemented through the tool described in 3.3. Poems with severe meter defects and missing rhymes are excluded from the finetuning dataset, resulting in less than 15% of the collected data being utilized.

Genres. In forming a corpus of poems for pretraining and finetuning, we did not limit its composition to any particular genre, style, or form, unlike many other works e.g. (Lo et al., 2022). As a result, the corpus contains, in addition to lyrics with different poetic meter*, tonality, and theme, also comic, satirical, and ironic poems, including a number of hard forms: pirozhki*, chastushka*, rubai*, limericks*, sonnets*, poems for children, poetic riddles, hymns (Greene et al., 2012, page 356), congratulations in verse etc.

The finetuning dataset comprises a total of 1,704,418 samples, distributed across various genres as follows: 52.7% lyrics, 24% hard forms, 11.9% humor and satirical poetry, 5% poems for children, and 6.4% others. The primary sources of poetry include:

- stihi.ru[5] (72%),
- poetory.ru[6] (2.8%),
- chitalnya.ru[7] (1.7%).

3.3 Accentuation and Poetry Scansion

For syllabo-tonic* poetry, the placement of stress marks follows specific rules for alternating stressed and unstressed syllables. Our algorithm supports five meters: trochee*, iamb*, dactyl*, amphibrach*, and anapest*. These five meters account for approximately 97% of all poems in the dataset. The remaining 3% include dolniks* and some exceptional cases (e.g., in the artishoki* genre).

For each stanza, the algorithm selects an optimal meter based on the reference sequence of stressed and unstressed syllables for the meter, the positions of ideal stresses, and whether these ideal stresses align with the permissible stress patterns of the words.

Russian pronunciation allows for variability in word stress, making automatic stress placement a computationally intensive task. The accentuator supports two main cases of variability: 1) certain phrases in the Russian language deviate from standard rules (there are several hundred of these phrases), 2) some words allow for variations in stress within the same grammatical form. To address this efficiently, the algorithm implements a beam search. For each line, there are two variants of stress placement, respectively, resulting in two clauzula* variants. The one that provides the best rhyme combination can be selected among these options.

[5]https://stihi.ru/

[6]https://poetory.ru/

[7]https://www.chitalnya.ru

4 Pretrained Models

The goal of this paper was to investigate whether using a character-level tokenizer and pretraining with it could improve automatic poetry generation. We hypothesized that character-level tokenization would represent text more accurately for poetry generation compared to byte-pair encoding. To test this hypothesis, we used two model architectures, which are described in detail below.

4.1 CharLLaMa

The CharLLaMa models follow the LLaMa architecture (Touvron et al., 2023). The only differences are: 1) character-level tokenization, 2) adjusted internal dimensions. We pretrain the models on the data described in 3.1. CharLLaMa is optimized to handle character-level tokenization and complex sequential patterns, aiming to outperform BPE-based models in capturing Russian poetry language structure. The initialization of the tokenizer vocabulary (a set of tokens for the tokenizer) was performed as follows: 1) the frequencies of Unicode symbols in the pretraining corpus were analyzed; 2) rare symbols with a frequency below 1000 have been excluded. This yields a vocabulary of 375 tokens, including special tokens `<s>`, `</s>`, `<pad>`, `<unk>`, and two special tokens for marking fragments of prose and poetry.

Two model variants were pretrained: 1.3B and 2.6B parameters (detailed specifications of the models are in Table 2).

Model training. CharLLaMa-1.3B model was pretrained over 14 days using 1 DGX 8*A100, leveraging CUDA V12.3.107 environment, and the CharLLaMa-2.6B was pretrained over 24 days using 1 DGX 8*H100 respectively. The learning parameters are listed in Table 7.

4.2 CharMamba

When using character-level tokenization, it's important to consider that it tends to make token sequences longer than methods like BPE or syllable-level tokenization due to the higher fertility[8] as shown in Table 8. Consequently, both model training and inference may take longer, in addition to

[8]Tokenization fertility was defined and analyzed for the BERT tokenizer in https://juditacs.github.io/2019/02/19/bert-tokenization-stats.html

increased memory consumption during autoregressive text generation and the time taken for generating text.

Given these limitations, we opted for the Mamba architecture as the second model for exploration, following the approach outlined in (Gu and Dao, 2023). Mamba is based on advancements in structured state space models, with an efficient hardware-aware design similar to FlashAttention, enabling it to be more efficient and faster that transformer-based models. We adopted the Mamba implementation from the official repository [9] and pretrain CharMamba on the dataset described in 3.1.

Model training. The CharMamba-1.3B model was pretrained over 5 days using 1 DGX A100 system with 8 GPUs, utilizing CUDA V12.3.107. The training parameters are detailed in Table 7.

4.3 Syllabo-tonic GPTs

Syllabo-tonic GPTs (stGPT) are based on the GPT-2 architecture (Radford et al., 2019), with modifications limited to tokenization and the size of hidden layers. We conducted experiments using two model variants: a 100M-parameter model (referred to as "stGPT small") and a 350M-parameter model (referred to as "stGPT medium"). Detailed specifications for both models are provided in Table 2.

Both models were pretrained on 3.1 and fine-tuned on 3.2 with hyperparameters listed in Table 7.

The tokenization algorithm for these models works as follows. First, the text is split into syllables, ensuring that each syllable contains exactly one vowel or consists of a single consonant (as in the case of certain prepositions and particles). Second, stressed syllables are marked using the "combining acute accent" symbol,[10] placed after the vowel. Third, the token sequences in each line of the poem are reversed from right to left, so that the last token of the line appears first, followed by the penultimate token, and so on. This reversal simplifies the model's task of selecting rhyming syllables during generation, reducing the likelihood of unsuccessful poem generation. Without this technique, the model might struggle to choose a rhyme that satisfies both lexical and grammatical constraints when reaching the end of a line. A similar approach has been used by Benhardt et al. (2018); Van de Cruys (2020).

5 Experimental Poetry Generation

5.1 Metrics

Evaluating generative LMs, especially for poetry, is challenging (Hämäläinen and Alnajjar, 2021) due to the lack of ground truth answers and the subjective nature of poetry evaluation. Both automatic tests and manual evaluations can assess poetry generation models. Poetry features strict structural requirements, such as syllabo-tonic forms that adhere to specific patterns of stressed and unstressed syllables. These elements can be verified algorithmically.

We introduced the metric **technicality**, calculated using the tool described in 3.3. A penalty is applied if the ideal meter requires an unstressed syllable, but the actual syllable in this position is stressed. More than two consecutive unstressed syllables are also penalized. A score of 0 indicates that the text does not match the typical patterns of syllabo-tonic poetry, while a score of 1 indicate a perfect match to a classic meter. Intermediate scores correspond to texts with varying numbers of defects; the closer the score to 1, the fewer the defects.

In addition to poetic meter, poems are typically expected to include rhyme. To evaluate the models' ability to generate rhymes, we measure the **rhyming level** as the proportion of quatrains with an ABAB rhyme scheme[*]. While this is a simplified approach — since generated poems may exhibit other rhyme schemes (e.g., ABBA, AABB, AABA) — the ABAB scheme is the most common in lyric poetry and represents the majority of samples in the training data.

Perplexity is a widely used automatic metric for evaluating the fluency of generated poems (Yan, 2016). It is calculated using a pretrained LMs. For our experiments, we used the ruGPT3-medium model[11] to compute perplexity. However, it is important to note that available LMs are typically trained on general-purpose text and may not fully capture the grammatical and stylistic nuances specific to poetry.

Out-of-vocabulary (OOV) rate is a simple metric used to detect abnormalities in generated poems. It measures the proportion of words in a text that do

[9] https://github.com/state-spaces/mamba

[10] https://unicodeplus.com/U+0301

[11] https://huggingface.co/ai-forever/ruGPT3-medium_based_on_gpt2

not appear in the finetuning dataset. A higher OOV rate indicates a greater likelihood of encountering unusual or nonsensical vocabulary in the generated text. The OOV rate is not a completely reliable indicator of vocabulary defects, as poetry generation is an open-ended task with no fixed dictionary. Lexical innovations, such as neologisms and creative word formation, are common in poetry. Poets often experiment with language boundaries, producing works like Lewis Carroll's "Jabberwocky" (Carroll, 2001) and its translations into Russian,[12] which consist of unconventional or invented words, or the Russian genre of "zaum,"[13]. However, in practice, "broken" vocabulary in generated poems often arises not from the model's creativity, but from a domain shift caused by finetuning language models like Mistral, ByT5, or ruGPT3-medium on poetic texts. This shift occurs because poetic language differs significantly from prose in terms of vocabulary, syntax, and the extensive use of figurative language. As a result, despite its limitations, the OOV rate is a simple and interpretable metric that provides a reasonable estimate of lexical defects.

Side-by-side human evaluation. A team of annotators evaluated the generated poems by comparing their outputs side-by-side with human-authored poems. Each annotator was given a prompt along with pairs of texts and instructed to select the text that best represented a poem in response to the given prompt. The criteria for comparing the texts, arranged in descending order of importance, were as follows:

- *Poeticness*: the text must be poetic and adhere to the rules of Russian syllabo-tonic versification.
- *Fluency, coherence, and meaningfulness*: the text must be free of grammatical errors and convey meaning.
- *Prompt relevancy*: the text must be relevant to the given prompt.

The prompts were generated using an LLM in a zero-shot setting, following a prompt schema similar to the one used for the finetuning dataset (3.2). For evaluation, we selected prompts suitable for poems with lengths ranging from 4 to 8 lines.

Author 1	Author 2	Num. of pairs
CharMamba-1.3B	human	873
CharLLaMa-1.3B	human	840
CharLLaMa-1.3B	CharMamba-1.3B	686

Table 1: Statistics of poem pairs used in the side-by-side evaluation study.

The total number of annotated pairs is 2,399, with detailed statistics provided in Table 1.

5.2 Experiments

Comparison with BPE models. In the first experiments, we evaluate the pretrained models listed in Section 4 and several foundation models with BPE tokenization: ruGPT3-large[14]; Mistral-7B-v0.1[15]; FRED-T5-1.7B[16]. All models were finetuned on the instruction dataset (subsection 3.2).

Syllabo-tonic tokenization. In the second part of the experiments, we examined the syllabo-tonic tokenization of the text. Tokens in this approach correspond to syllables, with separate tokens for stressed and unstressed syllables. This type of tokenization attempts to overcome the main limitation of character-level tokenization, which is the difficulty of capturing longer contexts. On average, syllables in the Russian language consist of approximately 2.3 letters, which aligns well with BPE tokenization.

We tested two models with 100M and 350M capacities, named “stGPT small” and “stGPT medium”. Table 2 presents the models' parameters. These models were pretrained on a dataset described in Section 3.1, then finetuned on the dataset described in Section 3.2. Training hyperparameters are presented in Table 7.

Table 4 shows the technicality scores for both human- and LM-authored poems across several genres.

Low-Rank Adaptation (LoRa). Full finetuning was used for all compared models. Our experiments with LLama 8B and LoRa demonstrated a significant degradation of the technicality of the generated poems, so we did not use this training option for the final comparison.

[12] https://prosodia.ru/catalog/stikhi/lyuis-kerroll-drug-moy-boysya-barmaglota/

[13] https://library.fiveable.me/key-terms/world-literature-ii/zaum

[14] ruGPT3-large is the Russian analog of the GPT-2 model, presented as a family of models of different sizes (Zmitrovich et al., 2024). The large version has 760M parameters

[15] Mistral-7B-v0.1 is the pretrained generative text model with 7 billion parameters proposed by the MistralAI team

[16] FRED-T5-1.7B (Zmitrovich et al., 2024) is the encoder-decoder pretrained model created for the Russian language

Model	$N\_positions$	N_embd	N_head	N_layer	$Num\_parameters$
stGPT small	1024	768	12	12	132,694,272
stGPT medium	1024	1024	16	24	365,840,384
CharLLaMa-1.3B	1024	1536	32	29	1,369,634,304
CharLLaMa-2.6B	2048	2064	24	28	2,641,199,664
CharMamba-1.3B	—	1320	—	31	1,238,287,360

Table 2: Model characteristics of the explored architectures. *N_positions* - number of positional embeddings, *N_emb* - token embedding size, *N_head* - number of transformer self-attention heads, *N_layer* - number of stacked decoder layers, *Num_parameters* — number of models parameters.

Model	**Sampling parameters**	**Technicality**	**Rhyming level**	**Perplexity**	**OOV rate**
stGPT medium	temp=0.9 top_p=0.75	0.72	0.467	70.28	0.004
stGPT small	temp=0.8 top_p=0.8	0.70	0.472	59.30	0.004
CharLLaMa-2.6B	temp=0.75 top_p=0.6	0.59	0.339	55.56	0.009
CharLLaMa-1.3B	temp=0.75 top_p=0.6	0.58	0.352	50.71	0.011
CharMamba-1.3B	temp=0.65 top_p=0.75	0.57	0.293	42.60	0.003
Mistral-7B-v0.1	temp=0.65 typical_p=0.75	0.57	0.192	72.41	0.012
FRED-T5-1.7B	temp=0.8 typical_p=0.7	0.26	0.126	**38.44**	**0.0029**
ruGPT3-large	temp=0.9 typical_p=0.7	0.06	0.002	45.59	0.0060
ByT5-large	temp=0.9 top_p=0.7	0.02	0.001	124.62	0.016
ByT5-small	temp=0.9 top_p=0.7	0.01	0.0	341.65	0.035
Human	n/a	**0.81**	**0.683**	72.81	0.0038

Table 3: Automatic metrics for models trained on the finetuning dataset (subsection 3.2). Lower *OOV rate* values indicate better performance, while higher values of *technicality* and *rhyming level* are preferred. *temp* in sampling parameters stands for temperature.

Author	**sonnets**	**rubai**	**limericks**	**chastushka**	**depressyashka**	**artishok**	**poroshok**
stGPT medium	**0.712**	0.596	0.613	0.689	0.591	0.361	0.578
stGPT small	0.699	0.559	0.636	**0.702**	0.533	0.266	0.567
CharLLaMa-2.6B	0.469	0.499	0.439	0.546	0.511	0.518	0.499
CharLLaMa-1.3B	0.495	0.522	0.484	0.568	0.504	0.487	0.496
CharMamba-1.3B	0.416	0.526	0.409	0.557	0.505	0.439	0.467
FRED-T5-1.7B	0.209	0.24	0.128	0.289	0.345	0.064	0.305
ruGPT3-large	0.059	0.063	0.048	0.079	0.132	0.031	0.064
Human	0.555	**0.644**	**0.644**	0.701	**0.64**	**0.88**	**0.642**

Table 4: Technicality scores for model- and human-authored poems across different genres. Higher technicality values indicate better performance.

Token-less models. We have also explored the performance of token-less models from ByT5 family (Xue et al., 2022). These models employ a tokenizer that operates at the byte level for utf-8 text encoding. It was expected that this tokenization approach would also allow the model to process individual characters of the text, thus helping the model acquire the Russian phonetics.

Finetuning with instructive samples. All samples for finetuning consist of an instructional prompt and poem text. For decoder models, that is, all except FRED-T5-1.7B, a special token separates the prompt and the poem. Samples were randomly combined into fixed-size batches with the right padding using a <pad> token. Prompt tokens were excluded from backpropagation in decoder models by setting an attention mask for each sample.

Experimental setup. The automatic metrics for all experiments were calculated uniformly according to the protocol described below.

- The CharLLaMa, CharMamba, and stGPT were trained from scratch according to the procedure described in Section 4, and subsequently trained on the finetuning dataset (subsection 3.2). Other models were trained only on the finetuning dataset. Models were finetuned using the transformers library v.4.36.2. The finetuning hyperparameters are described in the Appendix 7.

- To evaluate all the experiments and models, we use the test set of 1000 instructional prompts, each instructing to "Compose a quatrain about <theme>..." and being up to 200 characters long. All compared models were prompted to generate lyrics quatrains. If a model produced more than four lines, only the first four were considered. Per-genre evaluation was performed using 600 instructions following the format "Compose a poem in genre <genre> about <theme>".

- Nucleus sampling (Holtzman et al., 2019) was used as a generation algorithm for all models. For each prompt, a single sequence of tokens was generated and used as the result for evaluation. The sampling parameters were optimized for each specific model, with slight variations, as different models have distinct optimal configurations for these parameters.

6 Results

The results of the experiments are shown in Tables 3 for 1000 lyrics quatrains and Table 4 of Appendix refers for 600 generations of several other genres. The metrics indicate that poorly written poems can have lower perplexity, while human-authored poems have higher perplexity. As noted by Yi et al. (2018), it is essential to focus not only on the absolute value of perplexity but also on how well the obtained perplexity value fits within the range of values typical for works written by people. It can be helpful to approximate the corresponding distribution with a Gaussian distribution with a specific mean and variance.

The automatic evaluation results show that the fine-tuned ruGPT3-large and ByT5 models performed poorly in poetry generation, while Mistral-7B-v0.1 achieved better scores. However, Mistral-7B-v0.1's generated poems had higher perplexity and included many out-of-vocabulary words, likely due to its limited pretraining on Russian texts. Despite this, Mistral outperformed other models using BPE and byte-level tokenization, coming close to specialized character-level models. The FRED-T5-1.7B-based model performed slightly worse in terms of technical quality and rhyme but produced texts with fewer language errors, as shown by its lower perplexity and fewer out-of-vocabulary words.

Experiments with stGPT demonstrated that transformer models using syllable-level tokenization achieved the highest technicality scores among all models. For sonnets, stGPT even surpassed human-written poems in terms of technicality. However, this tokenization method has several limitations, as discussed in Section D. Additionally, while these models excel in technicality, they often produce texts with grammatical and fluency issues. These flaws do not affect technicality or rhyming metrics but reduce the overall quality of the poetry. Due to these limitations, we chose not to scale these models to a capacity comparable to CharLLaMa-1.3B, and no full-scale side-by-side evaluations were performed.

Human side-by-side evaluation results. We used expert side-by-side evaluations to assess poem quality, applying the Bradley-Terry model (Hunter, 2003) from the choix library[17]. This model was used to compare poems written by humans with

[17] https://github.com/lucasmaystre/choix

Author	Bradley-Terry Rate
Human	1.49
CharLLaMa-1.3B	0.23
CharMamba-1.3B	-0.20

Table 5: Bradley-Terry ratings for the compared models.

those generated by the models, as shown in Table 5.

Based on the side-by-side evaluation results, two key conclusions emerge: (1) automatic metrics alone are insufficient for a comprehensive and objective assessment of generative poetry, and (2) the significant gap between human-authored and generated poems suggests the need for further experimentation.

7 Conclusion

To summarize, our work focuses on generating Russian syllabo-tonic poetry across various genres and styles. We experimented with different approaches, such as character-level tokenization, using the CharLLaMa and CharMamba architectures. We extensively compared these character-level models with baseline models using various tokenization methods, finetuning them across datasets with different domain and rhythm structures. As part of our research, we created a new poetry spell-checking algorithm and accentuation system, which we have made available as open-source. Additionally, we released the top-performing pre-trained model for the Russian poetry generation. Finally, we propose poetry evaluation metrics and share insights on utilizing existing methods to assess the quality of generated poetry models.

Acknowledgments

We extend our sincere gratitude to the team of assessors for their professionalism and dedication in conducting side-by-side evaluations, which enabled the objective and reliable comparison of language models for Russian poetry generation. Special thanks to Timur Shentyurk, Danil Vyazovov, Vsevolod Alipov, Anastasiya Volodina, Igor Beloded, Mikael Desse, Evgeniy Sologub, and Anna Kostikova for their invaluable contributions.

We thank our colleagues Denis Shevelev and Valery Ternovsky for their insightful discussions on generative poetry and their valuable assistance.

We thank Leonid Sinev for his timely and meticulous assistance with references and LaTeX.

We also would like to especially thank Sergei Markov for his insightful suggestions and support, which significantly enriched this work and made it possible.

Limitations

This study has several significant limitations, which are discussed below.

Length of context. Although our generative LMs achieve solid results and promote state-of-the-art performance on various tasks, their context window size limits the model application on long-context tasks. The window size for CharLLaMa-1.3B and CharMamba-1.3B is 1024 tokens, and for CharLLaMa-2.6B, it is 2048 tokens. Remember that char tokenization imposes stricter limits on the number of words for processed sequences compared to models with BPE tokenization. The window context can include a much larger number of tokens, resulting in fewer words in the same context. However, poems are primarily short, and the context is not critical for them.

Speed and optimization. Longer token sequences in models with char-level tokenization lead to increased overhead (kv-cache for CharLLaMa models) and time for autoregressive inference compared to models with BPE tokenization. This is the tradeoff between the quality of poems and speed. The research regarding optimization has been left for future work.

Data biases. The generated poems are a result of the data used in the training. However, it's important to note that the study has limitations due to biases present in the training data, especially concerning Russian cultural aspects and copyright constraints. Because the data is culturally biased towards the Russian language, it cannot be directly applied to other languages.

New language models. New pretrained language models[18][19] and enhanced versions of the models[20] discussed in this paper are released frequently. The findings presented in Section 6 should not be generalized to these newer models, as modifications to the model architecture or pretraining pipeline may significantly impact their performance in generat-

[18]https://huggingface.co/yandex/YandexGPT-5-Lite-8B-pretrain

[19]https://huggingface.co/t-tech/T-pro-it-1.0

[20]https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.3

ing Russian-language poetry.

Ethical Consideration

Human creativity and possible misuse. Poetry is a form of creative expression and is often protected by copyright. AI-generated poetry should not infringe upon the rights of original creators. In our research, we only used licensed and open data for training. We must make efforts to avoid creating content that closely mimics or plagiarizes existing works. This helps maintain honesty and clarity in distinguishing between human and machine-generated art. We leave it to future work to address this issue.

Biases and data quality. Poetry is deeply rooted in cultural contexts. Understanding the cultural significance of certain themes, symbols, and language is crucial. The pretraining data for poetry generation of the presented models includes large segments from the internet domain and cultural specifics of Russian literature and cultural biases, consequently containing various stereotypes and biases. Therefore, such models are not transferable to other languages. We collected the datasets used to train poetry-generating AI to be diverse and representative of a wide range of poets and experiences. This helps to ensure that the output reflects a broad spectrum of human expressions. We understand that AI systems can unintentionally produce harmful content, such as violent, discriminatory, or otherwise inappropriate language. Ensuring that the poetry generated is free from such content is a key ethical responsibility.

Energy Efficiency and Usage. We compute the CO_2 emissions from pretraining and finetuning as Equation 1 (Strubell et al., 2019):

$$CO_2 = \frac{PUE * kWh * I^{CO2}}{1000} \qquad (1)$$

The power usage effectiveness (PUE) of our data centers is 1.3. The resulting CO2 emission values are CharLLaMa-1.3B — 837 kg, CharLLaMa-2.6B — 2008 kg, and CharMamba-1.3B — 732 kg, respectively. Model compression techniques and parameter-efficient finetuning methods can reduce the computational costs associated with model inference.

AI-assistants Help. We used Grammarly[21] and DeepSeek[22] to improve and proofread this paper, correcting grammatical, spelling, and style errors and paraphrasing sentences. As a result, some parts of our publication may be flagged as AI-generated or AI-edited.

We must consider ethical implications to ensure the responsible use of AI and respect for human creativity and culture. Developers and users of AI poetry tools should maintain responsible practices, honoring human creativity and the cultural significance of poetry.

References

Jonas Belouadi and Steffen Eger. 2022. Bygpt5: End-to-end style-conditioned poetry generation with token-free language models. *arXiv preprint arXiv:2212.10474*.

John Benhardt, Peter Hase, Liuyi Zhu, and Cynthia Rudin. 2018. Shall i compare thee to a machine-written sonnet? an approach to algorithmic sonnet generation. *arXiv preprint arXiv:1811.05067*.

Michele Boggia, Sardana Ivanova, Simo Linkola, Anna Kantosalo, and Hannu (TT) Toivonen. 2022. One line at a time - generation and internal evaluation of interactive poetry. In *International Conference on Innovative Computing and Cloud Computing*.

Lewis Carroll. 2001. *Jabberwocky and other poems*. Courier Corporation.

Tong Che, Yanran Li, Ruixiang Zhang, R Devon Hjelm, Wenjie Li, Yangqiu Song, and Yoshua Bengio. 2017. Maximum-likelihood augmented discrete generative adversarial networks. *arXiv preprint arXiv:1702.07983*.

Yanran Chen, Hannes Gröner, Sina Zarrieß, and Steffen Eger. 2024. Evaluating diversity in automatic poetry generation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 19671–19692, Miami, Florida, USA. Association for Computational Linguistics.

Michal Chudoba and Rudolf Rosa. 2024. Gpt czech poet: Generation of czech poetic strophes with language models. *arXiv preprint arXiv:2407.12790*.

Jonathan H Clark, Dan Garrette, Iulia Turc, and John Wieting. 2022. Canine: Pre-training an efficient tokenization-free encoder for language representation. *Transactions of the Association for Computational Linguistics*, 10:73–91.

Simon Colton and Geraint A Wiggins. 2012. Computational creativity: The final frontier? In *ECAI 2012*, pages 21–26. IOS Press.

[21] https://app.grammarly.com/

[22] https://chat.deepseek.com/

R Graeme Dunphy and Cristian Bratu. 2010. *The encyclopedia of the medieval chronicle*, volume 2. Brill Leiden.

Giorgio Franceschelli and Mirco Musolesi. 2024. On the creativity of large language models. *Preprint*, arXiv:2304.00008.

Emily Frith, Daniel B Elbich, Alexander P Christensen, Monica D Rosenberg, Qunlin Chen, Michael J Kane, Paul J Silvia, Paul Seli, and Roger E Beaty. 2021. Intelligence and creativity share a common cognitive and neural basis. *Journal of Experimental Psychology: General*, 150(4):609.

J. Fuller. 2017. *The Sonnet*. The Critical Idiom Reissued. Taylor & Francis.

Paul Fussell. 1979. *Poetic Meter and Poetic Form*. McGraw Hill, New York.

Albert Gatt and Emiel Krahmer. 2018. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, 61:65–170.

Roland Greene, Stephen Cushman, Clare Cavanagh, Jahan Ramazani, and Paul Rouzer. 2012. *The Princeton encyclopedia of poetry and poetics*. Princeton University Press.

Albert Gu and Tri Dao. 2023. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*.

Mika Hämäläinen and Khalid Alnajjar. 2019. Let's face it. finnish poetry generation with aesthetics and framing. *arXiv preprint arXiv:1910.13946*.

Mika Hämäläinen and Khalid Alnajjar. 2021. Human evaluation of creative nlg systems: An interdisciplinary survey on recent papers. *arXiv preprint arXiv:2108.00308*.

Mika Hämäläinen, Khalid Alnajjar, and Thierry Poibeau. 2022. Modern french poetry generation with roberta and gpt-2. *arXiv preprint arXiv:2212.02911*.

Haibo He and Edwardo A. Garcia. 2009. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284.

John Hollander. 2014. *Rhyme's reason: a guide to English verse*, 4th edition. Yale University Press, New Haven, CT.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*.

Jack Hopkins and Douwe Kiela. 2017. Automatically generating rhythmic verse with neural networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 168–178, Vancouver, Canada. Association for Computational Linguistics.

Zhiyuan Hu, Chumin Liu, Yue Feng, Anh Tuan Luu, and Bryan Hooi. 2024. Poetrydiffusion: Towards joint semantic and metrical manipulation in poetry generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18279–18288.

David R. Hunter. 2003. Mm algorithms for generalized bradley-terry models. *Annals of Statistics*, 32:384–406.

Mete Ismayilzada, Debjit Paul, Antoine Bosselut, and Lonneke van der Plas. 2024. Creativity in ai: Progresses and challenges. *Preprint*, arXiv:2410.17218.

Shakiba Kheradmand, Daniel Rebain, Gopal Sharma, Hossam Isack, Abhishek Kar, Andrea Tagliasacchi, and Kwang Moo Yi. 2023. Accelerating neural field training via soft mining. *Preprint*, arXiv:2312.00075.

Tatsuki Kuribayashi, Yohei Oseki, Takumi Ito, Ryo Yoshida, Masayuki Asahara, and Kentaro Inui. 2021. Lower perplexity is not always human-like. *arXiv preprint arXiv:2106.01229*.

Edward Lear. 2011. *Limericks*. Readers' & writers' genre workshop. Poetry. Benchmark Education Company.

Kai-Ling Lo, Rami Ariss, and Philipp Kurz. 2022. Gpoet-2: A gpt-2 based poem generator. *arXiv preprint arXiv:2205.08847*.

Tuan Nguyen, Phong Nguyen, Hanh Pham, Truong Bui, Tan Nguyen, and Duc Luong. 2021. Sp-gpt2: semantics improvement in vietnamese poetry generation. In *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1576–1581. IEEE.

Aleksandr Nikolaevich Nikolyukin. 2001. *Literaturnaya entsiklopediya terminov i ponyatii*. NPK Intelvak.

Petr Plecháč, Robert Kolár, Silvie Cinková, Artjoms Šeļa, Mirella De Sisto, Lara Nugues, Thomas Haider, Benjamin Nagy, Éliane Delente, Richard Renault, Klemens Bobenhausen, Benjamin Hammerich, Adiel Mittmann, Gábor Palkó, Péter Horváth, Borja Navarro Colorado, Pablo Ruiz Fabo, Helena Bermúdez Sabel, Kirill Korchagin, Vladimir Plungian, and Dmitri Sitchinava. 2023. PoeTree. Poetry Treebanks in Czech, English, French, German, Hungarian, Italian, Portuguese, Russian and Spanish.

Brian Porter and Edouard Machery. 2024. AI-generated poetry is indistinguishable from human-written poetry and is rated more favorably. *Scientific Reports*, 14(1):26133.

Maurilio De Araujo Possi, Alcione De Paiva Oliveira, Alexandra Moreira, and Lucas Mucida Costa. 2023. Carmen: A method for automatic evaluation of poems. In *2023 5th International Conference on Natural Language Processing (ICNLP)*, pages 244–247. IEEE.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Tatiana Shavrina and Olga Shapovalova. 2017. To the methodology of corpus construction for machine learning:"taiga" syntax tree corpus and parser. In *Proceedings of "CORPORA-2017" International Conference*, pages 78–84.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy. Association for Computational Linguistics.

Alexey Tikhonov and Ivan P Yamshchikov. 2018. Guess who? multilingual approach for the automated generation of author-stylized poetry. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 787–794. IEEE.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Tim Van de Cruys. 2020. Automatic poetry generation from prosaic text. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 2471–2480.

Olga Vechtomova, Gaurav Sahu, and Dhruv Kumar. 2020. Generation of lyrics lines conditioned on music audio clips. *arXiv preprint arXiv:2009.14375*.

M. Wachtel. 2004. *The Cambridge Introduction to Russian Poetry*. Cambridge Introductions to Literature. Cambridge University Press.

Yequan Wang, Jiawen Deng, Aixin Sun, and Xuying Meng. 2022. Perplexity from plm is unreliable for evaluating text quality. *arXiv preprint arXiv:2210.05892*.

Stanley Xie, Ruchir Rastogi, and Max Chang. 2017. Deep poetry: Word-level and character-level language models for shakespearean sonnet generation. *Natural Lang. Process. Deep Learn.*

Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. 2022. Byt5: Towards a token-free future with pre-trained byte-to-byte models. *Transactions of the Association for Computational Linguistics*, 10:291–306.

Rui Yan. 2016. i, poet: Automatic poetry composition through recurrent neural networks with iterative polishing schema. In *IJCAI*, volume 2238, page 2244.

Zhenyuan Yang, Zhengliang Liu, Jing Zhang, Cen Lu, Jiaxin Tai, Tianyang Zhong, Yiwei Li, Siyan Zhao, Teng Yao, Qing Liu, Jinlin Yang, Qixin Liu, Zhaowei Li, Kexin Wang, Longjun Ma, Dajiang Zhu, Yudan Ren, Bao Ge, Wei Zhang, Ning Qiang, Tuo Zhang, and Tianming Liu. 2024. Analyzing nobel prize literature with large language models. *Preprint*, arXiv:2410.18142.

Xiaoyuan Yi, Maosong Sun, Ruoyu Li, and Wenhao Li. 2018. Automatic poetry generation with mutual reinforcement learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3143–3153.

Chengyue Yu, Lei Zang, Jiaotuan Wang, Chenyi Zhuang, and Jinjie Gu. 2024. Token-free llms can generate chinese classical poetry with more accurate format. *arXiv preprint arXiv:2401.03512*.

Xingxing Zhang and Mirella Lapata. 2014. Chinese poetry generation with recurrent neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 670–680.

Xinran Zhang, Maosong Sun, Jiafeng Liu, and Xiaobing Li. 2023. Lingxi: A diversity-aware Chinese modern poetry generation system. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 63–75, Toronto, Canada. Association for Computational Linguistics.

Zhe Zhang, Karol Lasocki, Yi Yu, and Atsuhiro Takasu. 2024. Syllable-level lyrics generation from melody exploiting character-level language model. In *Findings of the Association for Computational Linguistics: EACL 2024*, pages 1336–1346.

Dmitry Zmitrovich, Aleksandr Abramov, Andrey Kalmykov, Vitaly Kadulin, Maria Tikhonova, Ekaterina Taktasheva, Danil Astafurov, Mark Baushenko, Artem Snegirev, Tatiana Shavrina, Sergei S. Markov, Vladislav Mikhailov, and Alena Fenogenova. 2024. A family of pretrained transformer language models for Russian. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 507–524, Torino, Italia. ELRA and ICCL.

Andrea Zugarini, Stefano Melacci, and Marco Maggini. 2019. Neural poetry: Learning to generate poems using syllables. In *Artificial Neural Networks and Machine Learning–ICANN 2019: Text and Time Series: 28th International Conference on Artificial Neural Networks, Munich, Germany, September 17–19, 2019, Proceedings, Part IV 28*, pages 313–325. Springer.

A Glossary

Acrostic is a poem or other word composition in which the first letter (or syllable, or word) of each new line (or paragraph, or other recurring feature in the text) spells out a word, message, or the alphabet. For more information see (Dunphy and Bratu, 2010, page 8).

Amphibrach is a metrical foot consisting of a stressed syllable between two unstressed syllables (Greene et al., 2012, page 31).

Anapest is a metrical foot consisting of two unstressed syllables followed by one stressed syllable (Greene et al., 2012, page 37).

Chastushka is a humorous quatrain with a simple rhyming scheme - see more details at (Nikolyukin, 2001, page 598).

Clauzula is the final part of a verse or stanza* starting from the last ictus* (Greene et al., 2012, page 141).

Dactyl is a metrical foot consisting of one stressed syllable followed by two unstressed syllables (Greene et al., 2012, page 179).

Dolnik is the type of poetic meter in Russian poetry, the peculiarity of which is a variable number of unstressed syllables between ictuses*. More information is available at (Nikolyukin, 2001, page 235).

Ictus is a stressed syllable (Greene et al., 2012, page 362).

Iamb is a metrical foot consisting of one unstressed syllable followed by one stressed syllable (Greene et al., 2012, page 360).

Limerick is a five-line poem with a rhyme scheme* AABBA, imitating the corresponding genre of English poetry (Lear, 2011).

Metrical foot is a regularly repeating pattern of 1 stressed and 1 to 2 unstressed syllables. There are two variants of disyllabic meter, called iambic* and trochee*, and three variants of trisyllabic meter, called amphibrach*, dactyl*, and anapest*. The main poetic meters that occur in training data are presented in Table 10.

Pirozhki, poroshki, depressyashki, artishoki are comic quatrains written without capital letters and punctuation marks, often with deliberate deviations from the rules of spelling. For each of these forms, there are strict constraints on the number of syllables, meter, and rhyme — see more details at the link

Poetic meter refers to the recurring pattern of stressed and unstressed syllables in lines of poetry. A comprehensive discussion of poetic meter and its nuances can be found in (Fussell, 1979).

Rhyme scheme describes which lines in a stanza* rhyme with each other, that is, contain the same or similarly sounding stressed endings of the lines (Hollander, 2014). Rhyme schemes presented in the finetune dataset (subsection 3.2) are listed in Table 11.

Rubai is a classical Persian poetry form, typically a quatrain with AABA or AAAA rhyming - see more details at (Greene et al., 2012, page 1227).

Stanza is a group of lines separated by blank lines from other stanzas. See (Greene et al., 2012, page 809) for more information.

Syllabo-tonic versification is based on 1) a fixed number of syllables in lines and 2) a regular pattern of stressed and unstressed syllables. In English-language literature, the term "accentual-syllabic" is more commonly used (Fussell, 1979, page 6), while "syllabo-tonic" is more common in scientific literature devoted to Slavic languages and Russian versification in particular (Wachtel, 2004). Given the specialization of this article on Russian-language poetry, we decided to use the "syllabo-tonic" variant.

Sonnet is a fixed verse poetic form consisting of 14 lines with constrained rhyming. For more information see (Fuller, 2017).

Trochee is a metrical foot consisting of one stressed syllable followed by one unstressed syllable (Greene et al., 2012, page 870).

B Fixing the spelling, punctuation, and tokenization issues

A significant portion of the training data was scraped from online sources, in particular from amateur poetry sites. The significant number of spelling and punctuation errors in these texts forced us to take special measures to clean the training data. A detailed description of the cleaning procedure is presented below.

We analyzed the collected poems described in section 3 for the most frequent misspellings and typos. As a result, many typos and common errors, which occurred up to 10 times in an 8 GB corpus, were corrected to their appropriate forms. Table 6 presents the 10 most frequent corrections. Based on this analysis, we created a “white list”, which served as the reference dictionary for identifying out-of-vocabulary words in the poetry corpus. We use dictionary-based replacements and heuristic rules for common spelling errors. When the algorithm detects a mistake, it checks if the correction exists in the reference dictionary and fixes it. We have developed about 30 rules based on regular expressions for this purpose. The typical problem cases are described below:

- Replace visually similar Latin characters with Cyrillic ones when they appear together in a word.
- Replace the combination of the letter "i" and the Unicode symbol U+0306 with the standalone Russian letter "j".
- To differentiate between Russian and English symbols, check for surrounding Cyrillic characters when dealing with single-letter words containing symbols from the character set [K, O, C, A, B, o, a, c, k, y].
- Replace various Unicode space characters[23] with the standard space character (U+0020).
- Handle cases where standard ASCII punctuation marks are replaced with full-width or half-width Unicode counterparts to convert them back to their ASCII prototypes.

The code implementing the above rules, along with all dictionary files, is publicly available as open source.[24]

One common issue in internet-sourced poetry texts is the presence of unnecessary commas. In generated poems, extra commas, especially between the subject and predicate, greatly reduce the quality of the text. To address this, we have implemented an algorithm that uses the perplexity of ruGPT3-medium[25] as an indicator of text likelihood. The algorithm functions by sequentially removing all commas from a sentence, except for the last one, and then comparing the perplexity of the sentence before and after each removal. If the perplexity significantly decreases after a comma is removed, that comma is deemed unnecessary and is eliminated. This method has the advantage of not requiring training on a specialized model. However, one drawback is that perplexity can be unreliable for short texts, as language models tend to consider shorter texts as less likely overall.

The above procedure affected about 10% of all collected data.

Defective text	Corrected text	Share, %
vraz	v raz	2.3
kak-budto	kak budto	2.0
gde to	gde-to	2.0
Kak-budto	Kak budto	1.6
kogda to	kogda-to	1.5

Table 6: The top frequent replacements in the corpus. The tokens are transliterated from Russian.

C Examples

Figure 1 presents a sample poem generated by our top model and its translation to English.

D Tokenizer Discussion

The use of language models with character-level tokenization is described in a number of papers (Belouadi and Eger, 2022; Yu et al., 2024). Compared to mainstream BPE tokenization and similar approaches, representing text at the character level makes it easier for the language model to handle poetry. For syllabo-tonic poetry, the key limitation lies in a strictly defined order of alternation of stressed and unstressed vowels (coinciding with syllables for the Russian language), as well as a certain number of syllables in each line. In BPE tokenization, different tokens contain different number of vowels. Therefore, the LM needs a more pretraining data to collect information about the composition of the tokens. In addition, taking into account vowel stress in the BPE scheme requires additional effort.

A compromise option can be considered a syllable-level representation of the text (Zugarini et al., 2019; Vechtomova et al., 2020). The disadvantage of this text representation is the difficulty of tokenizing for prose in some cases, for example, in multilingual contexts, when the syllabication rules differ for different languages.

Additionally, syllable-level tokenization, similar to BPE, performs poorly in some scenarios where

[23] In the texts collected on the Internet, nearly all the whitespace characters listed in the table https://www.unicode.org/Public/UCD/latest/ucd/PropList.txt are found

[24] https://github.com/Koziev/Spellchecker

[25] ai-forever/ruGPT3-medium_based_on_gpt2

```
Как быть счастливым, если нет покоя?          How to be happy if there is no peace?
Как быть счастливым, если в сердце мрак?      How to be happy if there is darkness in the heart?
Как быть счастливым, если нет прибоя?         How to be happy if there is no surf?
Как быть счастливым, если ты дурак?           How to be happy if you are a fool?

Как быть счастливым, если ты не знаешь,       How to be happy if you do not know,
Как быть счастливым, если ты не жил?          How to be happy if you have not lived?
Как быть счастливым, если ты не понял,        How to be happy if you do not understand,
Как быть счастливым, если не любил?           How to be happy if you have not loved?
```

Figure 1: The example of the generated poem. The English version is translated from the Russian.

Model	learning_rate	lr_scheduler_type	floating type	optimizer
CharLLaMa-1.3B	2e-5	constant	fp16	adamw_torch
CharMamba-1.3B	2e-5	linear	fp16	adamw_torch
CharLLaMa-2.6B	2e-5	constant	bf16	adamw_torch
stGPT small	5e-5	constant	bf16	adamw_torch
stGPT medium	5e-5	constant	bf16	adamw_torch
Mistral-7B-v0.1	2e-5	constant	bf16	adamw_torch
FRED-T5-1.7B	1e-4	constant	bf16	adafactor
ByT5-small	1e-4	constant	bf16	adafactor

Table 7: The hyperparameters of the models are in the finetuning stage for the experiments. The parameters were selected specially for each model.

the LM is required to understand the character-level composition of tokens, such as acrostics*.

Unfortunately, character-level tokenization has some disadvantages. They arise from the fact that token sequences are lengthened in comparison with BPE and syllable-level tokenization. Because of this, the time required for model pretraining and finetuning increases substantially. Memory consumption for the autoregressive text generation scheme and the time of this generation also increases.

Table 8 compares tokenization approaches for LMs described in Section 5.2.

E Pretraining Data Sources

The pretraining data is drawn from two sources: poetry and prose, with the proportion of each detailed in Table 9.

For prose, the following datasets were used as sources for the pretraining data:

- “YandexQ”[26] is a dataset of questions and answers scraped from Yandex.Q in the Internet domain. There are 836810 answered questions out of the total of 1297670.
- “Mail Question Answering” [27] is a set of question-answering pairs from real users.
- Instruction set of conversational agents [28] is a Russian instruction set of conversational domain.
- ruWikiHow[29] is a public dataset based on the parsed WikiHow source.
- Wikidepia[30] contains cleaned articles from Wikipedia dumps[31], one subset per language, each having a single train split. The Russian section was utilized for pretraining.
- Habr[32] is a dataset of posts and comments from habr.com[33], a Russian collaborative blog in the technical domain.

F Prompt Design

For every poem in the finetuning dataset (subsection 3.2), we create a synthetic prompt that varies in parameters (emotion, length, poetic meter*, etc.). The Russian example of the prompt for the creation

[26]https://huggingface.co/datasets/its5Q/yandex-q

[27]https://huggingface.co/datasets/Den4ikAI/mailruQA-big

[28]https://huggingface.co/datasets/Den4ikAI/russian_instructions_2

[29]Den4ikAI/ruWikiHow_instructions

[30]https://huggingface.co/datasets/wikimedia/wikipedia

[31]https://dumps.wikimedia.org/

[32]https://huggingface.co/datasets/IlyaGusev/habr

[33]https://habr.com/

Model	**Tokenizer**	**Characters per token w/o accentuation**	**Characters per token with accentuation**
ByT5	ByT5Tokenizer	0.56	0.55
CharLLaMa	CharacterTokenizer	1.00	1.00
Mistral-7B-v0.1	LlamaTokenizerFast	1.98	1.74
Llama-2	LlamaTokenizerFast	2.10	1.79
ruGPT3	GPT2TokenizerFast	3.20	2.12
FRED-T5-1.7B	GPT2Tokenizer	3.20	2.12
stGPT	StressedGptTokenizer	n/a	2.30

Table 8: Comparative results of tokenizers from the experiments described in Section 5.2. Characters per token is the default metric for tokenizer vocabularies of different sizes, obtained using the BPE and Unigram algorithms. N/A indicates cases where accentuation is required by design.

Type	Number of characters	Share, %
Prose	39,364,771,098	60.76
Poetry	25,427,281,242	39.24

Table 9: Statistics and proportion of prose and poetry texts in the pretraining dataset (subsection 3.1).

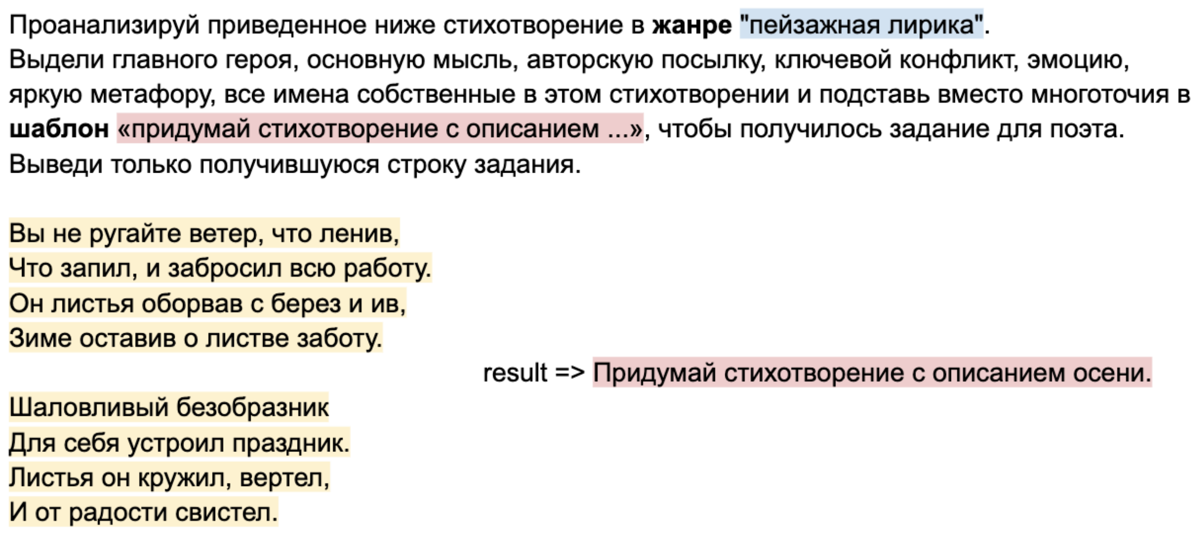
Meters	Share, %
iambic	57.88
trochee	34.28
amphibrachium	3.91
dactyl	2.24
anapaest	1.57
others	0.12

Table 10: The main poetic meters* and their proportions in the finetuning dataset (subsection 3.2).

Rhyming scheme	Share, %
`-A-A`	34.94
`ABAB`	34.68
`----`	16.09
`AABB`	11.26
`ABBA`	1.99
`A-A-`	0.55
`AABA`	0.4
others	0.9

Table 11: The most frequent rhyming schemes in the finetuning dataset.

of the synthetic prompt for the specific poem is presented in Figure 2.



Проанализируй приведенное ниже стихотворение в **жанре** "пейзажная лирика".
Выдели главного героя, основную мысль, авторскую посылку, ключевой конфликт, эмоцию, яркую метафору, все имена собственные в этом стихотворении и подставь вместо многоточия в **шаблон** «придумай стихотворение с описанием ...», чтобы получилось задание для поэта.
Выведи только получившуюся строку задания.

Вы не ругайте ветер, что ленив,
Что запил, и забросил всю работу.
Он листья оборвав с берез и ив,
Зиме оставив о листве заботу.

result => Придумай стихотворение с описанием осени.

Шаловливый безобразник
Для себя устроил праздник.
Листья он кружил, вертел,
И от радости свистел.

Figure 2: An example of the prompt and generated text. The yellow text represents a poem, while the red text denotes the TEMPLATE. The template is modified by the LLM based on its parameters and the analysis of the input poem, generating a instructive prompt for new poem creation.