Записки научных
семинаров ПОМИ
Том 546, 2025 г.

I. Bychkov, F. Chernogorskii, S. Averkiev, A. Fenogenova

EFFICIENT TOKENIZATION: BALANCING BABYMMLU, FERTILITY AND SPEED

Abstract. In Natural Language Processing (NLP), tokenization is a critical pre-processing step that significantly influences model performance. The choice of the tokenizer is crucial, especially given the contemporary situation with large LMs that are expensive to train. Our study investigates various subword-level tokenizers, considering their strengths and limitations. Based on our analysis, we propose a practical approach for comparing these tokenizers, considering factors such as tokenization effectiveness, vocabulary size, and tokenization speed. The paper reviews current tokenizer evaluation methods and contributes to a new evaluation dataset. Therefore, this paper aims to help researchers choose and train the most appropriate tokenizer for their tasks, especially when faced with limited training resources. Our objective is to empower the research community to make well-informed decisions about tokenizer selection and improve the quality of their language models.

§1. Introduction

Tokenization is an essential part of the language modeling creation pipeline. The choice of an optimal tokenizer can significantly affect the performance and training time of the language model (LM). Therefore, it is essential to select an effective tokenizer that ensures the quality of the model before the main training begins.

The direct way to compare tokenizers is to choose a tokenizer by retraining the whole model with each candidate. However, this method is resource-expensive and consumes much computational time. To address this problem, we propose a pipeline for comparing and evaluating tokenizers on models with fewer than 1 billion parameters in various domains and metrics before training the full model version.

The MMLU benchmark [13], commonly used to evaluate models in more than 50 domains, provides an extensive overview of a model's knowledge. However, it may not be suitable for small models requiring more tokens

Key words and phrases: NLP, LLM, tokenizer, tokenization, optimization, benchmark, dataset.

obtained during the pre-training stage. To address the issues of the small model's knowledge, we suggest using a specifically generated dataset and an estimation methodology that utilizes logit values to choose the correct answer.

We examine different types of subword-level tokenizers and conduct experiments on LMs trained with the considered tokenizers and on various frameworks. The combination of statistical metrics and a benchmark-like approach enables evaluation under restricted computational resources.

Thus, the contribution of our work can be summarized as follows:

- we contribute with the new BabyMMLU dataset[1] that is created specifically for small models evaluation;
- we present a methodology for evaluating tokenizers, ensuring a fixed experimental setup;
- we present a comparative analysis of various tokenizers, propose and publish an evaluation pipeline[2] for models under 1B parameters.

§2. Related Work

2.1. Tokenization. *Tokenization* transforms natural language text into a form that LM can handle. This transformation must be reversible for generative models to produce answers in natural language. LMs typically interpret text as a sequence of tokens, so the tokenizer needs to split the text sample into entities. Standard methods for selecting split borders include byte-level, char-level, word-level, and subword-level tokenization. Previous works have introduced a combination of these approaches [23] and more complex methods [25, 24]. In recent years, subword tokenization has become the most common approach.

2.2. Subword Tokenization. Subword tokenization methods, such as *Byte-pair encoding (BPE)* [9] and *Unigram LM* [10], are used to handle out-of-vocabulary units. BPE merges neighboring vocabulary units to create new tokens, while Unigram LM (introduced in the SentencePiece framework[3] [11]) uses a wide vocabulary and the expectation maximization algorithm to estimate unit probability. The initial vocabulary of BPE could contain Unicode bytes to avoid problems with unseen characters, and

[1] `https://huggingface.co/datasets/ai-forever/baby\_mmlu`

[2] `https://pypi.org/project/babymmlu`

[3] `https://github.com/google/sentencepiece`

the algorithm starts merging from the byte level. This method is called the BBPE algorithm, and only this algorithm is considered in further detail in the paper.

2.3. Pre-tokenization. The pre-tokenization scheme determines how text is split before being passed to the tokenizer. The choice of a pre-tokenization scheme can significantly impact model performance in different languages or domains. Previous studies [16, 17] have investigated the influence of tokenization of digits on arithmetic performance. In [15], authors have suggested that specific pre-tokenization schemes could improve model quality in the code domain.

2.4. Model Evaluation. Recently, there has been a growing interest in evaluating language models in alignment with human preferences. Although human evaluation is reliable, it is expensive and time-consuming. [18] introduced an approach that uses strong LMs as judges, achieving more than 80% agreement with human preferences. Classical benchmarking remains the essential and most common approach for model evaluation, with popular benchmarks such as GLUE [19], SuperGLUE [20], HELM [21] and MMLU [13] measuring model quality across various tasks. However, these benchmarks are mostly English-oriented, so we rely on the ruMMLU benchmark[4], a part of MERA [14], to evaluate Russian language models.

§3. Evaluation approaches

In further sections, existing metrics for tokenizer evaluation are reviewed. These metrics assess the quality of the tokenizer in different ways. The benchmark for small LMs (smaller than 1B parameters without embedding and output layers) evaluation is introduced, allowing the selection of the tokenizer from the corresponding LM quality perspective within a relatively small computation budget.

Our approach prioritizes computation budget economy, making the tokenizer selection process more effective and reducing associated costs. The pipeline enables you to measure various aspects, which can be helpful when selecting a tokenizer. Depending on the task, the pipeline user can prioritize what is most important for their specific industrial pipeline, such as

[4] `https://huggingface.co/datasets/ai-forever/MERA/viewer/rummlu/test`

benchmark metrics, speed, dictionary, etc. The methodology offers to accumulate all provided metrics in a final score, computed as a weighted average of all separate metrics. The selected example weights for the current industrial pipeline and all the measured results investigated are presented in Table 13 of Appendix L.

3.1. Benchmark-based.

MMLU and ruMMLU. The MMLU benchmark[5] assesses a language model's performance by presenting a prompt in a few-shot (5) multiple-choice format and requiring the model to select the answer (letter A-D) with the lowest cross-entropy.

The MMLU method works effectively for large models but could be more efficient for small ones. For example, according to the benchmarks MERA and OpenLLM leaderboard[6] results obtained by small models are not significantly better than random guessing (i.e., hovering around 25%).

This inefficiency is attributed to several reasons:

(1) It is challenging for small models to "understand" what is expected of them when different answers are presented directly in the prompt under different letters.
(2) Small models struggle to provide an answer to a question when the question is a complete sentence rather than simply continuing the question as if the answer were a natural continuation. Refer to Appendix K for a detailed analysis.
(3) The lack of knowledge and generalization capabilities hinders the model from producing a correct answer to the corresponding question.

Although it may be challenging to address the problems related to the completion of sentences and lack of knowledge without changing the dataset or model size, we can tackle the challenge related to the presentation of different answers (point 1). To address this, we can implement a cross-entropy-based approach to select the answer.

The process involves combining a question with each answer choice to calculate the cross-entropy of the model only on the answers while masking the original question. This results in obtaining a set of cross-entropy values for each token in the answer sentences. By summing up all the values for

[5] https://github.com/hendrycks/test

[6] https://huggingface.co/spaces/open-llm-leaderboard/open_llm_leaderboard

each sentence, we obtain the *total cross-entropy*. The answer with the lowest cross-entropy is selected. Dividing the obtained cross-entropy value by the number of characters or tokens in the answer provides two additional metrics: MMLU (*cross-entropy per character*) and MMLU (*cross-entropy per token*). These metrics, along with the total cross-entropy, help to evaluate the accuracy of the model.

BabyMMLU. A new benchmark called BabyMMLU has been developed to overcome the limitations of the MMLU benchmark. It consists of 2000 incomplete sentences, each offering four potential endings, with only one being correct. The average sentence length is 54.4 characters and the average ending length is 10.4. The dataset comprises sentences reflecting the general knowledge expected of a Russian middle-school student. It covers themes in history, science, music, literature, and other cultural-specific knowledge areas, including children's cartoons and common sayings.

The challenges highlighted in 3.1 are mitigated by the following aspects of BabyMMLU:

(1) Instead of evaluating the numerical position of the answer, the approach involves assessing the cross-entropy of each answer. The answer options themselves are not provided in the prompt.
(2) All the questions are presented in the format of sentence continuation.
(3) Both questions and answers are relatively short and do not cover complex or expert difficulty levels such as science or high school disciplines.

The dataset collection and annotation process was conducted by a team of professional editors and involved several stages:

- The editors initially created a list of candidate samples for the dataset.
- GPT-4 was used to assist in generating candidates for inclusion in the dataset. The specific prompt used for GPT-4 can be found in the Appendix D. Data obtained using this model account for approximately 20% of the total dataset.
- The editors manually filtered and selected the data.
- The final dataset was label-balanced to ensure that the correct answer choices had an equal probability of occurrence.

The resulting dataset example is presented in Figure 1 of the Appendix. The BabyMMLU benchmark simplifies the evaluation process through a

continuation format and the absence of domain-specific knowledge, providing a more convenient benchmark for evaluating small LMs.

BabyMMLU accuracy metrics are highly correlated with the MMLU results for larger models. These measurements are provided in Appendix J.

3.2. Metrics.

Bits per character. One of the standard metrics used for evaluating LLMs is *Bits per char* [30, 31, 32]. It shows the average number of bits needed to encode a character. To calculate it, the following formula can be used:

$$\mathrm{BPC}(w) = -\frac{1}{L}\sum_{i=1}^{N}\log_2 p(w_i|w_1,...,w_{i-1}) \tag{1}$$

Where L is the length of text in characters, N is the length of text in tokens, $w = \{w_1,...,w_N\}$ is sequence of tokens which represents the text, $p(w_i|w_1,...,w_{i-1})$ is the probability of token w_i given that the text before it consists of tokens $w_1,...,w_{i-1}$.

The following formula calculates the cross-entropy of the model on the given text:

$$\mathrm{CE}(w) = -\sum_{i=1}^{N}\log_e p(w_i|w_1,...,w_{i-1}) \tag{2}$$

Therefore, $\mathrm{BPC}(w) = \mathrm{CE}(w)/(L \cdot \ln 2)$.

To calculate the metrics for the entire dataset based on the formula above for a single text, we can use the following methods. *Macro-averaging*: for each text, calculate the value of *Bits per char* and then average this value across all texts. *Micro-averaging*: sum the cross-entropy values for all texts and divide by the total length of all texts.

Average characters per token. The tokenizer characteristic, the *average number of characters per token*, can be measured without the need to train a model. First, each text in the dataset must be tokenized, resulting in a list of tokens. The number of characters per token for a text is equal to the number of characters in the text divided by the number of tokens obtained as a result of tokenizing that text. To calculate the average number of characters per token for the dataset, one can also use micro- and macro-averaging.

Tokenization speed. The speed of tokenization depends on factors such as text length, batch size, operation type (encoding/decoding), dataset

characteristics, vocabulary size, hardware, framework (Hugging Face, SentencePiece), algorithm (BPE, Unigram) and tokenizer type (fast or regular). To assess tokenization speed, we fix these parameters, measure tokenization time several times, and then average the obtained values.

§4. Experimental setup

The proposed pipeline for training and evaluating tokenizers consists of the following stages:

(1) *Preparation of the dataset for tokenizer training.* This step is optional, as we have trained multiple tokenizer configurations on each dataset.
(2) *Training the tokenizer* on the prepared dataset.
(3) *Preparation of the dataset for model training* is performed only once; all models are trained on the same dataset.
(4) *Tokenizing the dataset for model training* using the trained tokenizer.
(5) *Model training* on the dataset from the previous step.
(6) *Evaluation* of model and tokenizer characteristics.

4.1. Training data.

Data for tokenizer training. In this study, we used open source datasets, namely FRW [26], RedPajama [27], StarCoder [28], as well as collected from web like Common Crawl[7], Wikipedia[8] and Stack Exchange[9]. For details on post-processing and cleaning the open source datasets, refer to their respective articles. Our datasets were filtered using established heuristics, including language-based filtering, removal of personal information, promotional content, and duplicates.

For training tokenizers, several datasets were prepared. All of these datasets were collected from sources in such a way that the proportions of text lengths taken from these sources were collected in accordance with the proportions of the potential training of the large model. The sizes of the datasets ranged from 30 to 300 billion characters. The proportions of the target sources are provided in Appendix A.

[7] `https://commoncrawl.org/get-started`
[8] `https://dumps.wikimedia.org/ruwiki/latest/`
[9] `https://archive.org/details/stackexchange`

Data for model training. One distinct dataset size of 30 billion characters with the same source proportions described above was prepared for model training. It was created to train models on a different set rather than tokenizers were trained on.

4.2. Tokenizers training. In the tokenizer training step, we explored different aspects such as the training framework (specifically SentencePiece and Hugging Face), tokenization algorithm (specifically BPE and Unigram), vocabulary size and the prepared dataset, along with its size. The complete list of all combinations of parameters is in Table 13 of Appendix L.

The vocabulary of trained tokenizers may vary between different training datasets. We are investigating the stability of vocabularies for different training datasets (see Appendix G).

4.3. Model training. This study uses LLaMA2 [12] models of different sizes. All model characteristics are identical, except for the vocabulary size. However, the vocabulary size only affects the size of the input embedding layer and the output linear layer. Therefore, the sizes of these layers are different, while the sizes of all other layers remain the same at 56M parameters. Note that the difference in model size does not create unequal conditions for evaluating models, as the evaluation is based on the same training time for the models. Additional details about the sizes of the trained models and their characteristics are listed in Appendix B.

Model comparisons were made, assuming that the models were trained for approximately 12 hours. This restriction was implemented to strike a balance between the duration of the experiments and the quality of the trained model. Consequently, the planned number of training batches, ranging from 50,000 to 160,000 batches, was selected to align with this time frame.

4.4. Evaluation. The experimental setup restricted the speed and timing of the evaluations. Checkpoints were assessed and saved every 3000 batches during training. After 12 hours of training, the resulting model checkpoint was saved and used further in the evaluation.

4.4.1. Benchmark-based.

MMLU and ruMMLU. Each benchmark was measured equally to calculate MMLU and ruMMLU using the answer letter and cross-entropy-based approach (see Subsection 3.1).

BabyMMLU. BabyMMLU demonstrates the model's overall performance in different fundamental areas of natural language understanding and reflects our desired LLM performance. BabyMMLU metrics were assessed after every 20 batches, and the average values for three BabyMMLU metrics (cross-entropy per character, cross-entropy per token, total cross-entropy) were calculated over an 11 to 12 hour training period. Then a weighted average of these three numbers was computed to obtain the final score.

As training neural networks is inherently stochastic, we conducted a thorough investigation of the dispersion of the resulting BabyMMLU metric values in several training iterations to provide a comprehensive understanding. The detailed results are provided in Appendix I.

4.4.2. Metrics.

Bits per character. The metric is highly correlated with the default LM loss but scales depending on the tokenizer compression rate to introduce context-length dependency. This metric was calculated according to the methodology described in Section 3.2.

Average characters per token. *Char per token* metric is used to measure the efficiency of context utilization. We calculated the average number of characters per token across the two groups of the evaluation datasets. The first group consists of English, Russian, and code data, the second group includes data in different languages as shown in Appendix F. The final results were obtained by averaging this measure within each group.

Tokenization speed. When examining the tokenization speed, we measured the tokenization time (in seconds) for each tokenizer five times and then averaged the values. The values of parameters we used when studying tokenization speed are listed in Appendix C.

§5. Results

The results of all experiments are presented in the Appendix L. The current section introduces the results presented in comparisons and sampled according to some axes of the parameter grid (vocab size, frameworks, algorithm, training dataset size). All the results presented below were obtained with the same computation budget.

5.1. Vocabulary size. *Vocabulary size* is the tokenizer's most transparent parameter. The tokenizer with a more extensive vocabulary could encode more characters with the same number of tokens. This leads to more

Vocab size	BabyMMLU	Char per token	Bits per char
42K	0.35	3.43	1.13
62K	0.36	3.56	1.11
90K	0.37	3.68	1.1
128K	**0.38**	3.77	**1.09**
192K	0.36	3.86	1.1
256K	0.35	**3.92**	1.12

Table 1. The comparison of tokenizers with the different vocabulary sizes. All of them were trained with the Huggingface framework and BPE algorithm on datasets with equal proportions of sources.

effective context utilization because the model can process more characters during one inference step.

As presented in Table 1 the model with medium vocabulary size achieves best quality on both *Bits per char* and BabyMMLU metrics. This outcome can be attributed, on the one hand, to the limited capacity of small tokenizers. On the other hand, models with a larger vocabulary require more computation time to construct an embedding space since there are more different tokens and each token appears less frequently in the training set.

Framework	Vocab size	Baby MMLU	Char per token	Char per token (out of domain)	Bits per char
HF	42K	0.353	3.433	1.797	1.131
SP		0.352	3.401	**1.891**	1.134
HF	62K	0.358	3.561	1.872	1.108
SP		0.36	3.553	**2.019**	1.111

Table 2. The comparison of training frameworks. BPE is the underlying algorithm in all cases, and datasets are the same.

5.2. Framework. In experiments with different training frameworks (Table 2), the tokenizers with the same vocabulary size achieve comparable results on almost every metric except *Char per token* in out-of-domain

data. This metric was evaluated on the dataset not used during training; it consists of several subsets from Wikipedia in different languages (Section F of the Appendix). This effect leads models that use the tokenizer trained with the SentencePiece framework to more efficient context utilization in case of further model training in new languages.

5.3. Algorithm. The comparison of training algorithms in Table 3 shows almost equal results for the candidates. However, the BPE algorithm achieves a higher compression rate on in-domain data. This effect may occur because Unigram tokenizers pay attention to the token's frequency in training data during the tokenization process, and even a slight shift in data sampling can significantly impact token distribution.

Algorithm	Vocab size	Baby MMLU	Char per token	Char per token (out of domain)	Bits per char
BPE	42K	0.353	**3.433**	1.797	1.131
unigram		0.36	2.944	1.778	1.129
BPE	62K	0.358	**3.561**	1.872	1.108
unigram		0.358	2.987	1.879	1.111

Table 3. Comparing tokenizers with HuggingFace training framework and different underlying algorithms.

5.4. Training data. The tokenizers, trained with the same parameters except for training data amount (from 30 to 300 billion characters), show equal results on almost every metric (Table 4). A significant difference was produced in terms of the *Char per token* metric on in-domain data for tokenizers trained with the Hugging Face framework. At the same time, SentencePiece tokenizers achieved comparable results. The larger training data set size leads to a lower compression rate and leads to overfitting for Hugging Face tokenizers (Appendix H).

§6. Conclusion

This paper addresses the critical challenge of selecting an optimal tokenizer for language modeling by proposing an efficient and systematic

Framework	Vocab size	Training data amount	BabyMMLU	Char per token	Bits per char
HF	42K	30	0.353	**3.433**	1.131
		150	0.355	3.386	1.119
		300	0.359	3.085	1.132
	62K	30	0.358	**3.561**	1.108
		300	0.362	3.197	1.108
SP	42K	30	0.352	3.401	1.134
		300	0.348	3.41	1.135
	62K	30	0.36	3.553	1.111
		300	0.364	3.552	1.11

Table 4. Evaluation of tokenizers trained with different amount of data. Training datasets were sampled with the same data proportions.

evaluation pipeline under limited resources. Our approach combines statistical metrics and benchmark-like evaluation to provide a comprehensive assessment, ensuring that the selected tokenizer improves model performance while managing computational costs. Our work introduces the BabyMMLU dataset and an evaluation technique, filling a significant gap in models under 1B params evaluation. We offer a resource-efficient framework for tokenizer evaluation and encourage further exploration and refinement of our methods to improve tokenizer selection in diverse modeling scenarios.

Limitations

Data Quality of the Pipeline. The effectiveness of the trained tokenizer and the trained LMs is highly dependent on the quality and size of the corpus used. A limited or biased corpus can lead to suboptimal tokenization and model performance, potentially missing critical linguistic nuances and specific domain cases, such as characters from formal or other languages.

The dataset suggested for evaluation may not encompass all potential use cases and domains, and the number of examples may not be sufficiently representative. This could result in models performing well on the benchmark but not on real-world tasks that deviate from the benchmark

scenarios. During the creation of the BabyMMLU test, we aimed to cover the most popular school fields and general knowledge.

Tokenizer Design and Constraints. The current methodology is designed to train small models under resource limitations and strict performance conditions. We assume that the tokenization process for small models will also apply to larger models. Training and evaluating language models, even small ones, can be computationally intensive. Limited computational resources restrict the ability to experiment with different architectures and hyperparameters, which affects the overall robustness of the approach. Small models trained on limited resources may only partially capture the complexity of the language, leading to inferior performance compared to larger models. This limitation could hinder the practical applicability of the results. Additionally, our paper explores only subword tokenization. The chosen granularity for tokenization (character-level, subword-level, word-level, etc.) may not be the optimal choice for all tasks or types of text, potentially impacting the performance of the language models, particularly in handling rare or out-of-vocabulary words. We only focus on subword-level tokenization in this research.

Future work. Language evolves, and the tokenizer and models may become outdated if not regularly updated with new data. This temporal relevance issue could affect the long-term applicability of the approach. The approach may be tailored to a specific language and might not transfer well to other languages without significant adjustments, which limits the methodology's cross-linguistic applicability. Our methodology does not cover these issues, and it is a part of further research.

Ethical considerations

Using efficient tokenizers can result in more effective use of computational resources, contributing to environmental sustainability and responsible use of technology. By prioritizing careful tokenizer selection, we reaffirm our commitment to ethical standards in AI development, instilling confidence in the creation of LMs.

The editors partly manually created the BabyMMLU dataset. Thus, it may contain various stereotypes and biases of the annotators. More detailed assessment is necessary to identify potential model vulnerabilities when generalizing to new and specific data.

Appendix A. Source ratios

Data sources and sampling proportions are shown in Table 5. These proportions were maintained in each dataset used for training tokenizers and models.

Source	**Language**	**Fraction**
FRW[10]	English	0.419
RedPajama[11] (arxiv)	English	0.021
RedPajama (book)	English	0.042
LibGen[12]	English	0.100
LibGen	Russian	0.110
Common Crawl[13]	Russian	0.128
Stack Exchange[14]	English	0.016
StarCoder[15]	Code	0.067
Wikipedia[16]	Russian	0.041
Wikiquote[17]	Russian	0.0001

Table 5. Data sources and sampling proportions. These proportions were maintained in every dataset used for training tokenizers and models.

Appendix B. Trained models

The relationship between model size, TFLOPs per forward pass, and the size of the model vocabulary is shown in Table 6.

Model training is carried out on a server with the following specifications: 8 GPU Tesla A100 80GB, 128 CPU-cores, 1944 GB RAM.

We use the llm-foundry[18] code base for model training. The optimizer used was the decoupled **AdamW** [29] with the cosine learning rate schedule. The learning rate was set to 3e-4 with a warm-up for the first 2000 batches, and the parameter $alpha\_f$ was set to 0.1.

The trained models had the following hyperparameters:

- Framework - Hugging Face

[18] `https://github.com/mosaicml/llm-foundry`

- Architecture - LlamaForCausalLM
- Model type - llama
- Intermediate size - 1800
- Hidden size - 600
- Number of hidden layers - 12
- Number of attention heads - 12
- Number of key-value heads - 12
- Hidden activation - silu
- Max position embeddings - 2048

Vocab size	**Number of parameters** (millions)	**TFLOPs**
259	56	1.06
42000	107	1.37
50257	116	1.43
52000	119	1.44
62000	131	1.52
90000	164	1.72
100264	176	1.8
128000	210	2.0
130944	213	2.02
192000	287	2.47
256000	363	2.95
512000	671	4.83

Table 6. The number of parameters and forward step speed for each model with particular vocab sizes.

Appendix C. Tokenization speed

The tokenization speed measurements are shown in Table 7. The experiments with tokenization speed were conducted on a single machine with 16 CPU cores 1500 MHz, 243 GB RAM. Each experiment consists of an encoding and decoding process of 10 MB of text data, and time for each part was obtained separately. Text data has been sampled from the dataset with the same frequency characteristics as in Appendix A.

Framework	Algorithm	Vocab size	Encoding	Decoding
HF	BPE	512K	9.05	**8.69**
HF	BPE	256K	9.05	<u>8.96</u>
HF	BPE	192K	**8.99**	9.09
HF	BPE	128K	9.13	9.19
HF	BPE	90K	<u>9.03</u>	9.48
HF	BPE	52K	9.09	10.08
HF	BPE	62K	9.05	10.27
HF	BPE	42K	9.2	10.68
HF	unigram	62K	12.16	11.65
HF	unigram	42K	12.29	11.66
HF	BPE	259	9.37	40.44
SP	BPE	52K	31.7	75.7
SP	BPE	62K	32.57	101.69
SP	BPE	42K	31.55	107.66

Table 7. The measurements of tokenization speed for 10MB of English texts in seconds. The measurements were obtained from N runs (where N is in range 5..120 for different rows) of encoding and decoding for texts sampled from real data. The best score is in bold, the second best one is underlined.

Question: Автором поэм «Илиада» и «Одиссея» является
The author of the epics 'Iliad' and 'Odyssey' is
Choices: Гесиод, Эсхил, Геродот, Гомер
(Hesiod, Aeschylus, Herodotus, Homer)
Answer: 4

Question: В двоичной системе счисления каждая цифра принимает одно из
(In the binary numbering system, each digit takes one of)
Choices: 10 значений, 8 значений, 2 значений, 16 значений
(10 values, 8 values, 2 values, 16 values)
Answer: 3

Figure 1. BabyMMLU samples. The dataset is provided in Russian, with English translations for clarification purposes.

> Generate 20 sentences in Russian. Each sentence should describe a fact (an object) from general world knowledge on the *topic*. The knowledge level should be the standard school curriculum up to 9th grade in a Russian school. The object should be located at the end of the sentence in such a way that it can be separated from the main part, but the sentence should remain whole. The sentence should be written in simple language and looks natural. Provide the answer as an array of sentences in JSON format with the fields 'text' (sentence without the object, if a colon or dash should be placed before the object in the full sentence, then put them. Also add a 'wrong' field, in which write 3 knowingly incorrect, wrong variants of continuation, they can be remotely related to the topic. Write only JSON and nothing else.

Figure 2. The prompt for synthetic data generation.

Appendix D. BabyMMLU creation

The example of the BabyMMLU dataset is presented in Figure 1.

The prompt for GPT-4, which was used to create the part of BabyMMLU, is presented in Figure 2.

The topic here is a placeholder for one of the educational subjects like "Mathematics", "History", "Geography", "Biology", "Physics", "Chemistry", "Literature", "Art", "Sports", "Computer Science".

Appendix E. Tokenizer training

In the experiments the Vocab Size, Framework, Algorithm and Amount of Training Data varied. At the same time, other parameters were maintained between experiments.

During the tokenizer training with HuggingFace framework, `BpeTrainer` with default parameters was applied. Special tokens `["<unk>", "<s>", "</s>"]` were added to represent the unknown token, the beginning and the end of the sequence, respectively. As a pre-tokenization step, numbers were separated into digits, and the whole text was transformed into bytes.

Some of the tokenizers were trained with the SentencePiece[19] framework provided by Google. The SentencePiece trainer was used with the default parameters except:

- `character_coverage`: 0.99995
- `allow_whitespace_only_pieces`: True
- `byte_fallback`: True
- `remove_extra_whitespaces`: True

Appendix F. Evaluation datasets

All the tokenizers were evaluated on the subsets of the training datasets of around 20 million characters.

Additionally, we measured the tokenizers in several foreign languages, including Armenian, Bashkir, Belarusian, Chinese, German, Italian, Japanese, Kazakh, French, Tatar, Ukrainian, Uzbek, and Yakut.

Appendix G. Vocabulary dispersion

The comparison of vocabularies for tokenizers. i-th row and column corresponds to the specific tokenizer. Each cell is the vocabularies intersection size. All the tokenizers have 42k vocab size and utilize BPE algorithm. The tokenizers in Table 8b were trained with the SentencePiece framework, whereas other tables consist of the tokenizers trained with HuggingFace. The tokenizers were trained with random sampled datasets of 30B characters except 8c, where the tokenizers were trained with larger datasets (300B characters).

Appendix H. Tokenizer training data amount

The analysis of the experiments in Table 4 shows that the tokenizers with smaller training data amounts achieve better results of Char per token among the tokenizers trained with the HuggingFace framework. To explore the influence of the training dataset size, the tokenizers with different vocabulary sizes were trained on different amounts of data, and Char per token was measured. Figure 3 shows the results that demonstrate performance degradation as the training data grows.

42000	41003	41043	41051	41049	41150	41039	40967
41003	42000	40989	40949	41003	40931	40935	40960
41043	40989	42000	41125	41068	41031	41089	40944
41051	40949	41125	42000	41058	41096	41119	40982
41049	41003	41068	41058	42000	41109	41093	41032
41150	40931	41031	41096	41109	42000	41138	40957
41039	40935	41089	41119	41093	41138	42000	41037
40967	40960	40944	40982	41032	40957	41037	42000

(a) 42K HF BPE

42000	40824	40945	40979
40824	42000	40608	40717
40945	40608	42000	40795
40979	40717	40795	42000

(b) 42K SP BPE

42000	41342	41456	41330
41342	42000	41450	41539
41456	41450	42000	41434
41330	41539	41434	42000

(c) 42K HF BPE 300 GC

Table 8. Size of the intersections of tokenizers' vocabularies. The tokenizers were trained independently. The average intersection sizes for setups 8a, 8b and 8c are: 41033.8, 40811.3, 41425.2 respectively.

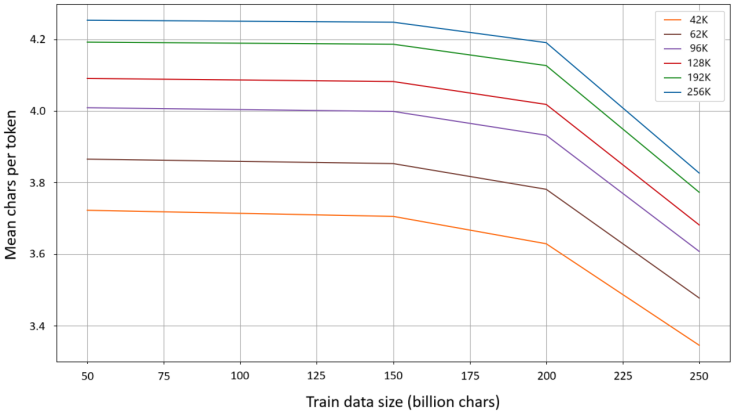

Figure 3. The *Char per token* results change over the tokenizer's training data growth. Every tokenizer was trained with the HuggingFace framework and BPE algorithm. The color reflects different vocabulary sizes. Each data point was obtained by averaging over 4 experiments.

Appendix I. Metric dispersion

In Table 9, we provide standard deviation values for each metric considered in our results. This table shows that metrics obtained in the experiments are stable, and the tables in Section 5 contain statistically significant results.

[19] `https://github.com/google/sentencepiece`

Framework	Algorithm	Vocab size	Dataset size (Gb)	Baby MMLU	Char per token	Char per token (out of domain)	Bits per char	MMLU
HF	BPE	42K	30	0.0047	0.0157	0.0133	0.0047	0.0016
HF	BPE	42K	300	0.009	0.0261	0.0101	0.0041	0.0007
HF	BPE	62K	30	0.0038	0.0164	0.0113	0.0053	0.0021
HF	BPE	62K	300	0.004	0.021	0.0184	0.0039	0.0019
HF	BPE	128K	30	0.0045	0.0148	0.0135	0.004	0.0016
SP	BPE	42K	30	0.0052	0.032	0.0464	0.0067	0.0015
SP	BPE	62K	30	0.0065	0.0197	0.0227	0.0065	0.0019

Table 9. The measurements of the metrics standard deviation for the tokenizers trained with the same procedure. The first four columns describe the tokenizer parameters, other columns are results of standard deviation measurements for each metric and were obtained by 7 different runs of the experiment.

Appendix J. BabyMMLU for bigger models

Model	BabyMMLU	ruMMLU	MMLU
llama2-7b[20]	0.615	0.452	0.453
mistral-7b-01[21]	0.686	0.676	0.601
llama3-8b[22]	0.706	0.696	0.666
llama2-13b[23]	0.682	0.563	0.548
mixtral-8x7b-instruct-v01[24]	<u>0.795</u>	**0.776**	**0.706**
llama2-70b[25]	**0.798**	<u>0.741</u>	<u>0.689</u>

Table 10. The accuracy measurements of open language models on MMLU [13], ruMMLU [14] and BabyMMLU benchmarks in single letter prediction setup.

Table 10 presents a comparison of the BabyMMLU benchmark with MMLU and ruMMLU on large open-source models. It shows that BabyMMLU is relatively efficient for large models, while the results of open-source LLMs remain consistent across benchmarks.

Appendix K. Methods for selecting answer

In Table 11, we provide the accuracy of a model trained with one of our best tokenizers (192K) for about 19 hours. The accuracy is provided on three datasets: MMLU continuation (a subset of the MMLU dataset with questions in continuation format), MMLU question (a subset of the MMLU dataset with completed questions), and BabyMMLU. We measured it with different prompt formats and several few-shot examples.

Measure	**Num of examples**	**Prompt format**	**MMLU continuation**	**MMLU question**	**Baby MMLU**
letters	0	12a	23.8	22.75	25.65
	5	12a	25.2	26.6	24.75
continuation	5	12b	27.7	**27**	30.3
	5	12c	27.5	25.9	32.25
	5	12d	27.3	25.9	33.45
	0	12d	**28.1**	25.9	**37.35**

Table 11. The comparison of evaluation format influence.

Appendix L. Experiments list

Table 13 shows the results of experiments with different tokenizer parameters, compared by different metrics. The *Final Score* value was obtained as linear combination of *Char per token*, *BabyMMLU* and *Bits per char* values scaled from 0 to 1 with weights 0.4, 0.2, 0.4 respectively.

Prompt	The term “budget deficit” refers to the A. annual increase in federal spending on the military B. amount of interest on the national debt C. difference between the initial budget proposals made by the president and Congress D. amount the government spends in excess of its revenues Answer:
Answer	D

(a) Answer is the single letter taken from the options.

Prompt	The term “budget deficit” refers to the A. annual increase in federal spending on the military B. amount of interest on the national debt C. difference between the initial budget proposals made by the president and Congress D. amount the government spends in excess of its revenues Answer:
Answer	amount the government spends in excess of its revenues

(b) Answer is the text of the right option.

Prompt	The term “budget deficit” refers to the Answer:
Answer	amount the government spends in excess of its revenues

(c) Answer is the text, options not included inside prompt.

Prompt	The term “budget deficit” refers to the
Answer	amount the government spends in excess of its revenues

(d) Answer is the natural continuation of the question.

Table 12. The prompt examples utilized for influence on MMLU and BabyMMLU results.

Framework	Algorithm	Vocab size	Dataset size (Gb)	Baby MMLU	Char per token	Char per token (out of dom.)	Bits per char	Final MMLU	Score
SP	BPE	62K	300	0.364	3.552	1.98	1.11	0.263	1.009
			30	0.36	3.553	2.019	1.111	0.268	1.006
			300	0.36	3.554	1.973	1.11	0.264	1.006
			30	0.359	3.545	1.971	1.109	0.264	1.006
			30	0.348	3.544	1.967	1.109	0.265	1.0
		42K	30	0.355	3.368	1.877	1.123	0.263	0.98
			30	0.352	3.401	1.891	1.134	0.262	0.978
			300	0.348	3.41	1.901	1.135	0.264	0.976
			300	0.347	3.411	1.897	1.135	0.261	0.976
			30	0.351	3.367	1.871	1.126	0.262	0.976
			30	0.344	3.404	1.895	1.134	0.261	0.973
			30	0.345	3.364	1.873	1.126	0.265	0.973
			30	0.34	3.371	1.877	1.126	0.261	0.971
HF	unigram	62K	30	0.358	2.987	1.879	1.111	0.261	0.947
		42K	30	0.36	2.944	1.778	1.129	0.26	0.937
	BPE	256K	30	0.348	**3.921**	**2.258**	1.12	0.267	1.034
		192K	30	0.363	3.863	2.187	1.097	0.268	1.045
		128K	30	**0.382**	3.768	2.057	**1.09**	0.266	**1.048**
		90K	30	0.366	3.675	1.963	1.098	**0.269**	1.027
		62K	30	0.359	3.561	1.878	1.107	0.261	1.008
		62K	30	0.358	3.561	1.872	1.108	0.265	1.007
		62K	300	0.362	3.197	1.857	1.108	0.265	0.972
		62K	300	0.359	3.198	1.843	1.106	0.264	0.971
		42K	30	0.355	3.386	1.753	1.119	0.262	0.983
		42K	30	0.353	3.396	1.774	1.119	0.262	0.983
		42K	30	0.353	3.433	1.797	1.131	0.263	0.982
		42K	30	0.35	3.396	1.778	1.119	0.264	0.982
		42K	30	0.351	3.385	1.755	1.12	0.264	0.981
		42K	30	0.35	3.398	1.778	1.12	0.262	0.981
		42K	30	0.349	3.396	1.781	1.119	0.262	0.981
		42K	30	0.347	3.395	1.789	1.119	0.264	0.98
		42K	30	0.344	3.396	1.778	1.119	0.263	0.978
		42K	30	0.343	3.398	1.772	1.12	0.259	0.978
		42K	30	0.343	3.396	1.783	1.121	0.26	0.977
		42K	30	0.34	3.433	1.793	1.132	0.263	0.975
		42K	300	0.359	3.085	1.736	1.132	0.264	0.949
		42K	300	0.35	3.085	1.735	1.133	0.263	0.944
		42K	300	0.346	3.085	1.736	1.132	0.263	0.943
		42K	300	0.339	3.036	1.717	1.125	0.264	0.937
		42K	300	0.339	3.037	1.716	1.125	0.262	0.937
		42K	300	0.335	3.036	1.717	1.124	0.263	0.935
		42K	300	0.334	3.036	1.717	1.125	0.263	0.934
		259	30	0.32	0.877	0.658	1.264	0.259	0.657
		259	30	0.302	0.877	0.658	1.264	0.254	0.648

Table 13. The results of experiments with different tokenizer parameters, compared by different metrics.

References

1. A. Aho and J. Ullman, *The Theory of Parsing, Translation and Compiling*, Prentice-Hall, 1972.
2. American Psychological Association, *Publications Manual*, American Psychological Association, 1983.
3. A. Chandra, D. Kozen, and L. Stockmeyer, *Alternation*. — Journal of the Association for Computing Machinery **28** (1981), 114–133.
4. G. Andrew and J. Gao, *Scalable Training of L1-Regularized Log-Linear Models*, in: Proceedings of the 24th International Conference on Machine Learning, 2007, pp. 33–40.
5. D. Gusfield, *Algorithms on Strings, Trees and Sequences*, Cambridge University Press, 1997.
6. M. Rasooli and J. Tetreault, *Yara Parser: A Fast and Accurate Dependency Parser*, Computing Research Repository, arXiv:1503.06733 (2015). Available: `http://arxiv.org/abs/1503.06733`. (version 2)
7. R. Ando and T. Zhang, *A Framework for Learning Predictive Structures from Multiple Tasks and Unlabeled Data*. — Journal of Machine Learning Research **6** (2005), 1817–1853.
8. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser, and I. Polosukhin, *Attention Is All You Need*, in: Advances in Neural Information Processing Systems **30**, 2017.
9. R. Sennrich, B. Haddow, and A. Birch, *Neural Machine Translation of Rare Words with Subword Units*, ArXiv preprint arXiv:1508.07909 (2015).
10. T. Kudo, *Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates*, ArXiv preprint arXiv:1804.10959 (2018).
11. T. Kudo and J. Richardson, *SentencePiece: A Simple and Language Independent Subword Tokenizer and Detokenizer for Neural Text Processing*, ArXiv preprint arXiv:1808.06226 (2018).
12. H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, *et al.*, *Llama 2: Open Foundation and Fine-Tuned Chat Models*, ArXiv preprint arXiv:2307.09288 (2023).
13. D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt, *Measuring Massive Multitask Language Understanding*, ArXiv preprint arXiv:2009.03300 (2020).
14. A. Fenogenova, A. Chervyakov, N. Martynov, A. Kozlova, M. Tikhonova, A. Akhmetgareeva, A. Emelyanov, D. Shevelev, P. Lebedev, L. Sinev, *et al.*, *MERA: A Comprehensive LLM Evaluation in Russian*, ArXiv preprint arXiv:2401.04531 (2024).
15. G. Dagan, G. Synnaeve, and B. Rozi?re, *Getting the Most Out of Your Tokenizer for Pre-Training and Domain Adaptation*, ArXiv preprint arXiv:2402.01035 (2024).
16. R. Nogueira, Z. Jiang, and J. Lin, *Investigating the Limitations of Transformers with Simple Arithmetic Tasks*, ArXiv preprint arXiv:2102.13019 (2021).

17. A. Thawani, J. Pujara, P. Szekely, and F. Ilievski, *Representing Numbers in NLP: A Survey and a Vision*, ArXiv preprint arXiv:2103.13136 (2021).
18. L. Zheng, W. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. Xing, *et al.*, *Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena*, in: Advances in Neural Information Processing Systems **36**, 2024.
19. A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman, *GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding*, ArXiv preprint arXiv:1804.07461 (2018).
20. A. Wang, Y. Pruksachatkun, N. Nangia, A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman, *SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems*, in: Advances in Neural Information Processing Systems **32**, 2019.
21. P. Liang, R. Bommasani, T. Lee, D. Tsipras, D. Soylu, M. Yasunaga, Y. Zhang, D. Narayanan, Y. Wu, A. Kumar, *et al.*, *Holistic Evaluation of Language Models*, ArXiv preprint arXiv:2211.09110 (2022).
22. J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. Casas, L. Hendricks, J. Welbl, A. Clark, *et al.*, *Training Compute-Optimal Large Language Models*, ArXiv preprint arXiv:2203.15556 (2022).
23. J. Tiedemann, *Character-Based PSMT for Closely Related Languages*, in: Proceedings of the 13th Annual Conference of the European Association for Machine Translation, 2009.
24. P. Koehn and K. Knight, *Empirical Methods for Compound Splitting*, ArXiv preprint cs/0302032 (2003).
25. S. Nie?en and H. Ney, *Improving SMT Quality with Morpho-Syntactic Analysis*, in: COLING 2000 Volume 2: The 18th International Conference on Computational Linguistics, 2000.
26. G. Penedo, Q. Malartic, D. Hesslow, R. Cojocaru, A. Cappelli, H. Alobeidli, B. Pannier, E. Almazrouei, and J. Launay, *The RefinedWeb Dataset for Falcon LLM: Outperforming Curated Corpora with Web Data, and Web Data Only*, ArXiv preprint arXiv:2306.01116 (2023). Available: `https://arxiv.org/abs/2306.01116`.
27. Computer, T., *RedPajama: An Open Source Recipe to Reproduce LLaMA Training Dataset*, 2023. Available: `https://github.com/togethercomputer/RedPajama-Data`.
28. R. Li, L. Allal, Y. Zi, N. Muennighoff, D. Kocetkov, C. Mou, M. Marone, C. Akiki, J. Li, J. Chim, *et al.*, *StarCoder: May the Source Be with You!*, ArXiv preprint arXiv:2305.06161 (2023).
29. I. Loshchilov and F. Hutter, *Decoupled Weight Decay Regularization*, 2019.
30. T. Lei, *When Attention Meets Fast Recurrence: Training Language Models with Reduced Compute*, 2021. Available: `https://arxiv.org/abs/2102.12459`.

31. R. Al-Rfou, D. Choe, N. Constant, M. Guo, and L. Jones, *Character-Level Language Modeling with Deeper Self-Attention*, 2018. Available: https://arxiv.org/abs/1808.04444.
32. K. Hwang and W. Sung, *Character-Level Language Modeling with Hierarchical Recurrent Neural Networks*, 2017. Available: https://arxiv.org/abs/1609.03777.

SberDevices

E-mail: ivankrylatskoe@gmail.com

E-mail: fechernogor@gmail.com

E-mail: averoo@gmail.com

E-mail: alenush93@gmail.com

Поступило 28 февраля 2025 г.