
Improving Large Language Model Russian Adaptation with Preliminary Vocabulary Optimization

M. M. Tikhomirov^{1,*} and D. I. Chernyshev^{1,**}

¹*Lomonosov Moscow State University, Moscow, Russia*

Received March 1, 2024

Abstract—Most of Large Language Model (LLM) text comprehension capabilities come from generative pre-training on large corpora which includes texts of different domains, languages and tasks. As a consequence the LLM performance in a specific language depends on its representation in the training data which for most state-of-the-art models was biased towards English language. The issue is commonly alleviated by further pre-training on the target language, however, due to limited model capacity this often results in knowledge forgetting and text understanding degradation. We argue that the performance drop can be avoided by employing parameter-efficient tuning methods that preserve the integrity of the original model. In this work, we investigate the effectiveness of different vocabulary optimization and adapter tuning schemes for LLM Russian adaptation. Our experimental results with Solar-10.7B LLM show that language adaptation process can be substantially accelerated by transferring the embeddings from smaller language-tuned counterparts. Moreover, we find that preliminary vocabulary optimization stabilizes further adapter-tuning thus improving target language generalization. By applying our two-stage language adaptation approach we obtain state-of-the-art results on Russian Super Glue and MMLU-RU language understanding datasets for sub-30B parameter open-source LLMs.

2010 Mathematical Subject Classification: 68T50

Keywords and phrases: *Large Language Models, Tokenization, Language Adaptation*

1. INTRODUCTION

Large language modeling (LLM) has seen rapid development since the introduction of ChatGPT. Thanks to large-scale generative pre-training and continuous instruction tuning ChatGPT is capable of solving many text-based tasks in zero-shot manner on a level comparable to specialized task-tuned models. However, despite the public availability ChatGPT and other state-of-the-art LLM API-based solutions aren't applicable to data-sensitive environments that due to increased security risks prohibit document processing outside of local networks. On the other hand, the current performance of local LLM counterparts is substantially lower in zero-shot setting for most of downstream tasks and thus their adoption for target applications often requires additional training to reach human acceptable quality. This issue becomes more noticeable for non-English domains that due to underrepresentation in popular pre-training datasets [11] are processed with much lower quality [3].

Developing a language-specific LLM isn't feasible in low-resource environments as the computational costs of the pre-training procedure are several times higher than fine-tuning on downstream tasks [2]. As a more cost-efficient alternative, researchers [1, 3] explored the possibility of improving target-language comprehension through partial training of existing state-of-the-art multilingual LLM. While direct LLM tuning¹²³ on instructions of the target language enables better language utilization this approach retains tokenization inefficiency and thus the computational performance

* E-mail: tikhomirov.mm@gmail.com

** E-mail: chdanorbis@yandex.ru

¹ <https://github.com/avocardio/Zicklein>

² <https://github.com/22-hours/cabrita>

³ <https://github.com/IlyaGusev/rulm>

drops. Vocabulary optimization [1, 3] addresses the issue by reducing tokens-per-word ratio while also improving semantic informativeness of tokenization units. Continued pre-training was also shown [3] to be beneficial for target-language comprehension, however, the quality gains heavily depend on the model size and could be marginal for smaller LLMs (i.e. 7B parameters). At the same time even best continued pre-trained LLMs proved [12] to be less efficient than the target-language counterparts trained from the scratch.

Our hypothesis is that irregular performance improvements of continued pre-training is the result of knowledge forgetting caused by overfitting during intermediate-layer tuning. We argue that the issue could be alleviated with two-stage scheme of embedding warm-up and subsequent continued pre-training. While the two-stage approach was originally shown [3] to be inferior to end-to-end tuning we find that its efficiency can be enhanced by appropriate embedding initialization. Moreover, we show that post-training optimization such as LoRA merging weight modulation can revert knowledge forgetting and is more efficient after vocabulary optimization. By applying our two-stage approach to SOLAR 10.7B Russian adaptation we achieve state-of-the-art results on Russian Super Glue and MMLU-RU language understanding benchmarks for sub-30B parameter open-source LLMs.

2. RELATED WORK

Language adaptation of large language models (LLM) has been a popular alternative to full pre-training since the introduction of billion parameter models whose training costs can exceed tens of PF-days [2].

Lakew [4] et al. were one of the first to show that adaptation of input vocabulary to unseen languages paired with embedding layer tuning could be more efficient than training a full neural machine translation model from scratch. This observation was confirmed by Kuratov et al. [5] who improved the multilingual BERT performance on Russian language by completely substituting the vocabulary and training only the new embeddings. A detailed study of vocabulary optimization was conducted by Rust et al [6], which concluded that the performance improvements generalize to all languages. Yang et al. [7] explored the vocabulary extension method to alleviate the XLM-R low performance on rare languages and achieved substantial improvements for Mongolian and Tibetan. To cut the computational costs of vocabulary optimization approach for monolingual models Zeng et al. [8] proposed bilingual lexicon embedding initialization, which allows skipping the embedding layer tuning stage for encoder-only pre-trained models.

The vocabulary optimization was extended to LLM. Vries et al. [9] directly applied the approach to GPT-2 Italian adaptation and achieved a comparable results to the variant trained from the scratch. Tikhomirov et al. [1] complemented the approach with morphologically accurate tokenization and showed that beside language comprehension improvements vocabulary optimization significantly boosts computational efficiency and task generalization capabilities of LLM. Similarly, Cui et al. [3] utilized vocabulary extension approach to reduce the training costs of LLM Chinese adaptation however they found the technique less efficient than vocabulary substitution for lower model sizes as language comprehension didn't improve without intermediate layer tuning.

3. METHODOLOGY

Our approach improves on the previous work [1] and can be summarized in the following steps:

1. Build a new Unigram tokenization vocabulary on target language corpus,
2. Rebuild the embedding and LM head layers to account the new vocabulary size,
3. Initialize the updated layers,
4. Continued target-language pre-training.

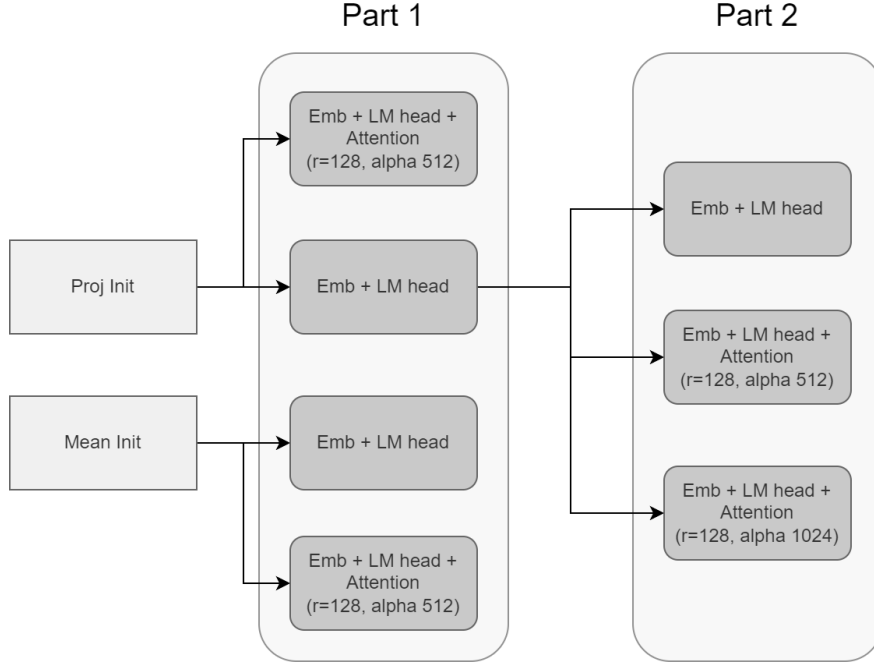


Figure 1. Experimental Scheme

3.1. Tokenization

We used Unigram as the tokenization algorithm, since past experiments [1] showed that it is better suited for the Russian language than BPE. Tokenization was trained using the sentencepiece package⁴.

3.2. Initialization

3.2.1. Mean initialization. A common embedding initialization [5–7] strategy is to leverage existing embeddings by aggregating the overlapping positions overlapping with the new vocabulary. The simplest implementation averages the embeddings the tokens that were assigned to new vocabulary units by the original tokenization algorithm:

$$\begin{aligned} v_{new}(\tilde{t}_i) &= \frac{1}{K} \sum_{j=1}^K v_0(t_j); \\ tokenize(\tilde{t}_i) &= [t_1, \dots, t_K], \end{aligned} \quad (1)$$

where $tokenize$ is original tokenization function, $\tilde{t}_i$ - token in new tokenization, t_i - token in original tokenization, $v_0(\dots)$ - the token embedding in the original model.

3.2.2. Projection initialization. In this work, we explored the possibility of initializing new LLM embedding matrices using a previously adapted smaller counterpart. Assume we have two models $small_base$ and $large_base$ of the same neural architecture that have different number of Transformer layers but the same hidden_size. Assume there is a linear transformation W_{proj} that maps the embeddings of smaller model V_{small_base} to the larger counterpart V_{large_base} :

$$\begin{aligned} V_{large_base} &= V_{small_base} W_{proj}; \\ W_{proj} &\approx V_{small_base}^{pinv} V_{large_base}; \end{aligned} \quad (2)$$

Then the spatial updates for embeddings and lm head matrices of a adapted small model $small_adapt$ can be transferred to currently trained $large_adapt$:

⁴ <https://github.com/google/sentencepiece>

$$V_{large_adapt} \approx V_{small_adapt} W_{proj}; \quad (3)$$

For a *small_adapt* model we use previously adapted Mistral-7B using vocabulary optimization approach [1], and for *large_base* we consider Solar-10.7B, a layer-scaled version Mistral-7B that was further pre-trained on multi-lingual corpora.

3.3. Continued target-language pre-training

For continued pre-training we follow the scheme used by Cui et al. [3]. Similarly, we employ LoRA parameter efficient tuning to training the model on generative pre-training task on the new corpora. To ensure proper knowledge integration we tune embeddings directly and use LoRA for all linear blocks, which is attention $W_{k_proj}, W_{v_proj}, W_{q_proj}, W_{o_proj}$, post-attention MLP $W_{gate_proj}, W_{up_proj}, W_{down_proj}$, and output embeddings W_{lm_head} .

We hypothesize that due to knowledge discrepancy between newly initialized embeddings and intermediate layers continued pre-training in End-to-End fashion will lead to instabilities and knowledge biasing. To test the hypothesis we split the training process into 2 stages: embedding warm-up (i.e. vocabulary optimization) and full continued pre-training. Both schemes are compared in evaluation section.

4. EXPERIMENTS

4.1. Training Data

For our adaptation experiments, we collected documents from the following domains:

- Russian and English Wikipedia
- Pikabu
- News
- Habrahabr
- Stackoverflow
- Books

The documents were deduplicated using the Locality Sensitive Hashing Minhash algorithm. To ensure grammatical and semantic coherence of training examples we removed metadata, links, and comment sections of web-pages from the texts. UTF-8 normalization was used to remove non-standard symbols like emoji or logograms, restricting the vocabulary to Cyrillic and Latin languages. The text structure was standardized to single space characters (i.e. no multi-spaces or multi-newlines). The total size of the pre-training dataset is 40 GB.

To improve example sampling we have changed the original example grouping method implemented in the Transformers library⁵. The approach is similar to doc-sentences from the work on RoBERTa [10], where paragraphs are used instead of sentences. Each document was divided by a line break character and all samples were constrained to start either from the beginning of the document or from the beginning of a paragraph. All text remainders were discarded as they often exhibited distorted contextual meaning. To better pack the example batches on our hardware (and thus, achieve maximal computational efficiency) each sample was limited to 512 tokens.

Our pilot experiments found that embedding warm-up stage needs at least 20GB of data to fully restore the embedding information flow. Therefore for two-stage continued pre-training scheme we split our data into 2 parts of 20 GB each and the same domain distribution statistics.

⁵ <https://github.com/huggingface/transformers>

4.2. Implementation details

4.2.1. Hardware. Training on the Russian language modeling dataset took place on 24 Nvidia V100 (12 nodes with 2 GPUs on each). Training on Russian Super Glue and all evaluation took place on RTX 4090.

4.2.2. Software. Multi-node training was carried out using DeepSpeed Zero 1. We are used Pytorch 2.1.2, transformers 4.37.2, PEFT 0.6.0.

4.2.3. Training Parameters. Block size 512, total batch size 128, learning rate 2e-5 and 5e-5 in case of LoRA, mixed-precision training with fp16 and O3. For all experiments we use LoRA with decomposition matrix rank $r = 128$.

4.3. Experimental Scheme

A diagram of all the experiments we conducted is presented in Figure 1. First we tested the efficiency of embedding initialization strategies on the first part of data which was reserved for embedding warm-up. Then using the best performing embedding warm-up checkpoint we continue pre-trained the model on the second half of pre-training data. This scheme is denoted as **Two-Stage**. The **End-to-End** baseline was trained on both parts using the best embedding initialization using the same training settings but extended learning rate scheduler to account for increased epoch.

5. RESULTS

5.1. Evaluation

We evaluate language modeling performance on three datasets: Russian Super Glue, MMLU-RU (**Translated**)⁶ and MMLU-RU (**MERA**)^[14].

MMLU (Massive Multitask Language Understanding) are benchmarks covering various categories such as mathematics, physics, philosophy, history and others. The Translated MMLU-RU version was obtained by translating the original dataset via Yandex.Translate API and thus exhibits translation artifacts such as wrong homonymy resolution and transliteration of domain-specific terms. While the latest iteration of MMLU-RU presented in MERA^[14] benchmark is much smaller it was designed to address this issue and as a result contains only high quality human written language understanding tests. All MMLU-RU evaluations were conducted in five-shot setting using the official prompt format.

Russian Super Glue (RSG) benchmark ^[15] consists of 9 natural language understanding tasks. To maintain consistency with the existing submissions we used evaluation protocol from Saiga model github repository⁷. All models were fine-tuned on mix of training data of downstream tasks and then evaluated with default parameters and repetition penalty 1.0.

5.2. Initialization Impact

5.2.1. Vocabulary optimization We explored the embedding initialization efficiency for vocabulary optimization for both pre-training schemes on the first part of pre-training dataset. The result of embedding warm-up stage can be seen in Figure 2. As can be seen from the validation plots the projection initialization has a significantly higher convergence rate achieving the final value of mean initialization at the 5000th step while maintaining a consistent loss margin.

Since mean embedding initialization was previously shown ^[3] to perform better in End-to-End setting for vocabulary extension we tested whether it's the case for our complete vocabulary substitution approach. The comparison between both projection strategies is illustrated in Figure 3 and Figure 4. Contrary to the previous observations ^[3], two-stage training is superior in the case of mean initialization as End-to-End model demonstrates a lower validation performance than embedding warm-up. However, the embedding warm-up stage can be fully skipped with projection initialization which in case of End-to-End training has the best validation performance of all first part models.

⁶ https://github.com/NLP-Core-Team/mmlu_ru

⁷ <https://github.com/IlyaGusev/rulm>

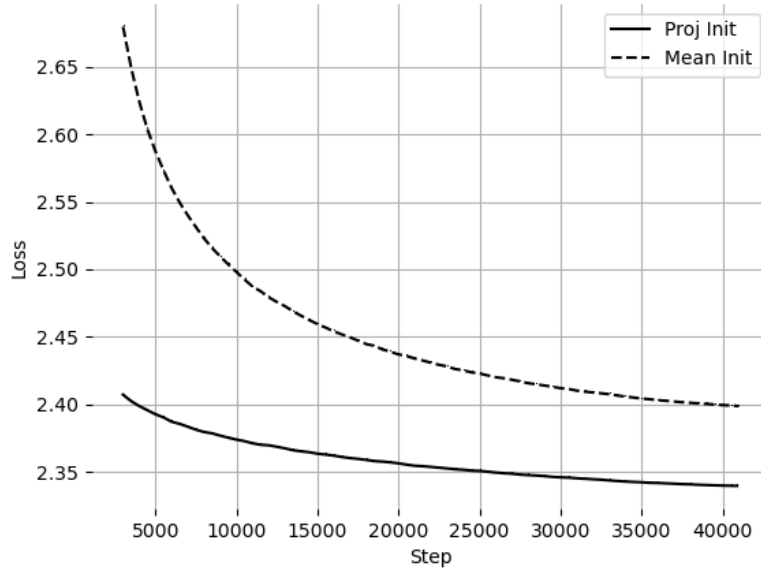


Figure 2. Loss with different initialization

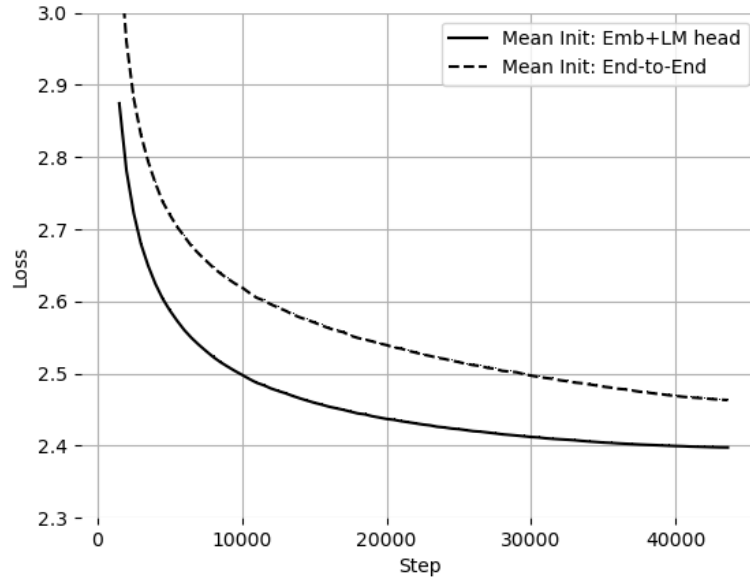


Figure 3. Loss difference for two additional training variants after mean initialization

5.3. Continued Pre-Training

Since the first stage determined the superiority of projection initialization all models (including End-to-End) in this section follow this embedding warm-up strategy. Beside comparing Two-stage and End-to-End schemes we also investigate the necessity of intermediate layer-tuning.

We trained 3 different configurations on second half of dataset:

- 1) Training only embedding and LM head

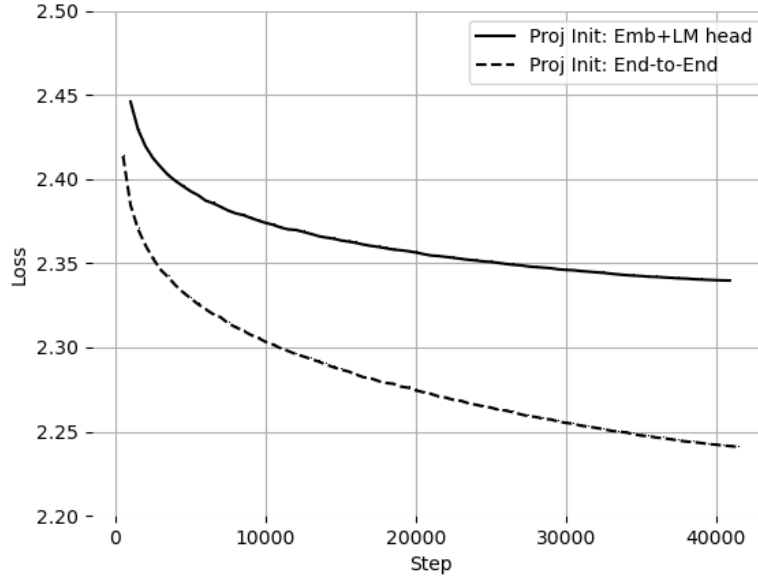


Figure 4. Loss difference for two additional training variants after projection initialization

- 2) Training with attention matrices and alpha 1024 and
- 3) Training with attention matrices and alpha 512.

The evaluation results are reported in Table 1. As can be seen, further training of embeddings and LM head on the second half of the pre-training data has inconsistent results and barely increases the quality of the model. However, all variants with intermediate layer tuning exhibit significant quality degradation.

5.4. Post-training Merging Factor Tuning.

The value of α in LoRA tuning has a direct impact on convergence as it scales the applied learning rate coefficients. In our case the learning rate was multiplied by a factor $\frac{\alpha}{r} = \frac{512}{128} = 4$. Increasing the α had a negative effect on performance which suggests substantial knowledge forgetting. On the other hand, LoRA is an approximation of cumulative gradient update and thus by altering α after the training we can still control the knowledge update in the original model.

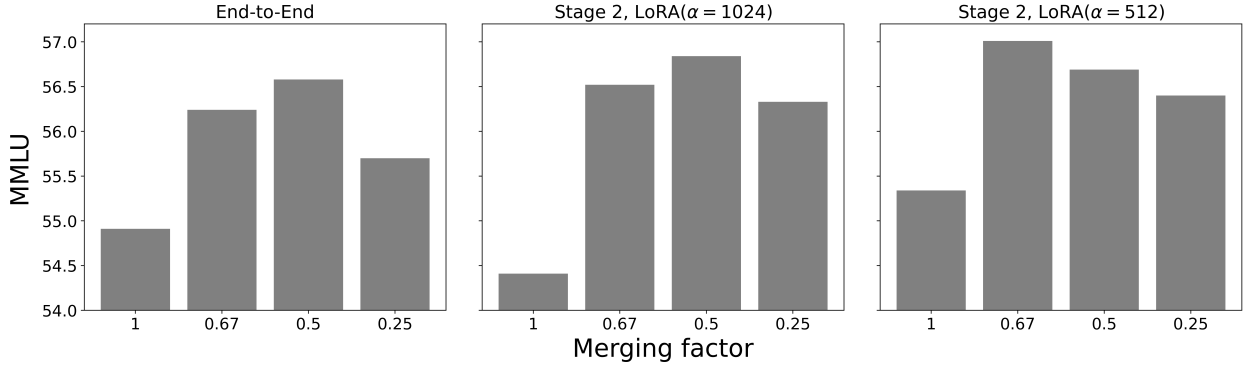
We tested 4 different α factors for post-training merging: 1, 0.67, 0.5, 0.25 and applied it to all attention matrices. Results can be seen in Figure 5 and in the Table 1. We found that modulating the merging factor, not only reverts to the original knowledge, but also retains new insights. Reducing the factor to 0.5 substantially boosts the language performance of all LoRA tuned models. Moreover, training with the lower α is not equivalent to merging with reduced factor as the model trained with $\alpha = 1024$ and merged with factor of 0.5 substantially outperforms the counterpart that was trained with $\alpha = 512$ from the start while also achieving state-of-the-art results for RSG benchmark. At the same time applying merging factor 0.67 to LoRA tune with $\alpha = 512$ closes the gap to a comparable level and reaches state-of-the-art results for MMLU-RU MERA benchmark among sub-30B parameter open-source LLMs.

6. CONCLUSION

In this work, we investigated the possibility of alleviating the knowledge forgetting during LLM language adaptation with two-stage pre-training. First we tested the performance of various embedding initialization strategies for vocabulary optimization. To speed-up the convergence we proposed projection initialization which transfers the embedding knowledge of smaller LLM to

Table 1. Evaluation results on benchmarks

	MMLU-RU		
	Translated	MERA	RSG
SOLAR 10.7B	57.9	0.708	0.794
Two-Stage			
+ Vocabulary optimization			
Projection init	54.5	0.690	0.791
Mean init	54.6	0.706	0.790
+ Continued pre-training			
Embedding-only	55.6	0.684	0.792
LoRA($\alpha = 1024$)	<u>54.4</u>	0.690	0.796
+ merging factor $\frac{1}{2}$	<u>56.8</u>	0.719	0.805
LoRA($\alpha = 512$)	<u>55.3</u>	0.693	0.798
+ merging factor $\frac{2}{3}$	57.0	0.722	0.800
End-to-End			
LoRA($\alpha = 512$)	54.9	0.695	0.787
+ merging factor $\frac{1}{2}$	56.5	0.714	0.794

**Figure 5.** The effect of LoRA adapter merging factor on model performance

larger counterparts. In contrast to previous works we found conventional mean initialization the most efficient in two-stage scheme. At the same time, our projection initialization outperforms mean initialization in both Two-stage and End-to-End pre-training strategies. Moreover the End-to-End model with projection initialization had the fastest convergence rate and the best validation results, thus allowing bypassing the embedding warm-up stage entirely. However, despite better initialization and LoRA-tuning models that were trained with intermediate layer-tuning exhibited significant performance instabilities potentially caused by knowledge forgetting. To alleviate the issue we proposed post-training merging factor tuning. By reducing the merging factor we substantially boosted the performance of continued pre-trained models of both Two-Stage and End-to-End schemes. By tuning the merging factor we achieved State-of-the-art results on Russian Super Glue and MMLU-RU MERA benchmarks for sub-30B parameter open-source LLMs.

Acknowledgement

The research was carried out using the equipment of the shared research facilities of HPC computing resources at Lomonosov Moscow State University. The work of Mikhail Tikhomirov was supported by Non-commercial Foundation for Support of Science and Education "INTELLECT". The work of Daniil Chernyshev was supported by Non-commercial Foundation for Support of Science and Education "INTELLECT".

REFERENCES

1. M. Tikhomirov, and D. Chernyshev, "Impact of Tokenization on LLaMa Russian Adaptation", arXiv preprint arXiv:2312.02598 (2023).
2. T. Brown, et al., "Language models are few-shot learners", Advances in neural information processing systems, Volume 33, pp.1877-1901, 2020.
3. Y. Cui, et al., "Efficient and Effective Text Encoding for Chinese LLaMA and Alpaca", arXiv preprint arXiv:2304.08177. – 2023.
4. S. Lakew, et al., "Transfer Learning in Multilingual Neural Machine Translation with Dynamic Vocabulary", in Proceedings of the 15th International Conference on Spoken Language Translation, 2018, pp. 54–61.
5. Y. Kuratov, M. Arkhipov, "Adaptation of deep bidirectional multilingual transformers for Russian language", arXiv preprint arXiv:1905.07213. – 2019.
6. P. Rust, et al., "How Good is Your Tokenizer? On the Monolingual Performance of Multilingual Language Models", in Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), 2021, pp. 3118–3135.
7. Z. Yang, et al., "CINO: A Chinese Minority Pre-trained Language Model", Proceedings of the 29th International Conference on Computational Linguistics, 2022.
8. Q. Zeng, et al., "GreenPLM: Cross-Lingual Transfer of Monolingual Pre-Trained Language Models at Almost No Cost", arXiv preprint arXiv:2211.06993. – 2022.
9. W. Vries, M. Nissim, "As Good as New. How to Successfully Recycle English GPT-2 to Make Models for Other Languages", in Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, 2021, pp. 836–846.
10. Y. Liu et al., "Roberta: A robustly optimized bert pretraining approach", arXiv preprint arXiv:1907.11692. – 2019.
11. H. Touvron, et al., "LLaMA: Open and Efficient Foundation Language Models", arXiv:2302.13971., 2023.
12. Y. Huang et al., "C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models", Advances in Neural Information Processing Systems, Volume 36, 2024.
13. J. Devlin, M. W. Chang, K. Lee and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language", Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1, pp. 4171-4186, 2019.
14. A. Fenogenova et al., "MERA: A Comprehensive LLM Evaluation in Russian", arXiv preprint arXiv:2401.04531. – 2024.
15. T. Shavrina, et al., "RussianSuperGLUE: A Russian language understanding evaluation benchmark", arXiv preprint arXiv:2010.15925. – 2020.