Computational Linguistics and Intellectual Technologies:
Proceedings of the International Conference "Dialogue 2026"

June 24–26, 2026

LLMTF: An Open Framework and Benchmark for Evaluating Large Language Models in Russian

Mikhail Tikhomirov **Evgenii Zlunikin** **Denis Grigoriev** **Daniil Chernyshev**
Lomonosov Moscow State University, Research Computing Center
Moscow, Russia
tikhomirov.mm@gmail.com, zlunick@mail.ru,
dagrig14@yandex.ru, chdanorbis@yandex.ru

Abstract

The rapid development of Large Language Models (LLMs) requires reliable and scalable evaluation tools. However, the research community faces significant fragmentation, outdated frameworks designed for foundational models, and closed proprietary benchmarks. These problems are particularly noticeable in the Russian-language segment, which suffers from a lack of high-quality datasets and open tools. To address these challenges, we introduce LLMTF[1], a novel open framework and benchmark for evaluating LLMs. LLMTF unifies the evaluation process through a message format, supports various backends (vLLM, HuggingFace, OpenAI API), and offers a comprehensive benchmark comprising 55 datasets across 5 key categories: knowledge, skills, long context, RAG, and instruction following. Additionally, it features an LLM-as-a-Judge Arena with style control for robust open-ended evaluation, and integrates specialized benchmarks for complex tasks like extreme long-context summarization and encyclopedic generation. Our framework provides research groups with an accessible and local tool to effectively evaluate models, identify their strengths and weaknesses, and analyze the impact of changes or model modification processes on generation quality.

Keywords: Large Language Models, Evaluation, Benchmark, Russian Language, Open-Source
DOI: 10.29003/2075-7182-2026-24-627-643

LLMTF: Открытый фреймворк и бенчмарк для оценки больших языковых моделей на русском языке

Михаил Тихомиров **Евгений Злуникин**
Денис Григорьев **Даниил Чернышев**

Московский государственный университет имени М.В. Ломоносова,
Научно-исследовательский вычислительный центр
Москва, Россия
tikhomirov.mm@gmail.com, zlunick@mail.ru,
dagrig14@yandex.ru, chdanorbis@yandex.ru

Аннотация

Стремительное развитие больших языковых моделей (LLM) привело к необходимости создания надежных и масштабируемых инструментов для их оценки. Однако исследовательское сообщество сталкивается со значительной фрагментацией датасетов, устаревшими фреймворками, разработанными для базовых моделей, и закрытыми проприетарными бенчмарками. Эти проблемы особенно остро проявляются в русскоязычном сегменте, который страдает от нехватки качественных датасетов и открытых инструментов. Для решения этих проблем мы представляем LLMTF — новый открытый фреймворк и бенчмарк для оценки LLM. LLMTF унифицирует процесс оценки через формат сообщений (message format), поддерживает различные бэкенды (vLLM, HuggingFace, OpenAI API) и предлагает комплексный бенчмарк, состоящий из 55 датасетов в 5 ключевых категориях: знания, навыки, длинный контекст, RAG и следование инструкциям. Кроме того, он включает в себя LLM-as-a-Judge арену с контролем стиля для оценки ответов на открытые вопросы, а также в бенчмарк интегрированы специализированные бенчмарки для

[1] https://github.com/RefalMachine/llmtf_open

сложных задач, таких как суммаризация экстремально длинных текстов и генерация энциклопедических статей. Наш фреймворк предоставляет исследовательским группам доступный и локальный инструмент для эффективной оценки моделей, выявления их сильных и слабых сторон, а также анализа влияния изменений и процессов модификации моделей на качество генерации.
Ключевые слова: Большие языковые модели, Оценка качества, Бенчмарк, Русский язык

1 Introduction

The rapid development of Large Language Models (LLMs) has led to the need for reliable and scalable evaluation tools. Currently, the research community faces several challenges in this area. First, there is significant fragmentation: there are numerous disparate datasets, implemented across various codebases, which complicates unified testing. Second, many popular evaluation frameworks were developed during the era of foundational models and were not originally optimized for modern instruction-tuned models and message-based interaction formats (message format). Third, the most comprehensive and modern benchmarks are often closed proprietary systems designed exclusively for evaluating frontier models from major technological corporations.

These problems are particularly noticeable in the Russian-language segment. The Russian language is characterized by a shortage of high-quality datasets and open-source developments. Existing solutions, such as the MERA benchmark(Fenogenova et al., 2024), have several significant drawbacks: they contain outdated tasks, requires loading the results into an external system to obtain final scores, and have long been unable to effectively separate models of different quality.

At the same time, alongside large corporations, there are many small research teams whose goal is to improve existing models (e.g. adapting them to the Russian language) or to develop specialized compact LLMs. For such teams, having a local, fast, and convenient evaluation tool is critical. In the context of rapid iterative development, where hundreds of intermediate checkpoints need to be analyzed, using commercial API solutions becomes economically unviable and slow. In this scenario, the problem of benchmark contamination takes a back seat to the need for fast and local comparison of different versions of the same model.

To address these problems, We present LLMTF - a novel open framework and benchmark for evaluating the quality of large language models. LLMTF unifies the evaluation process through a message format, supports various backends (vLLM, HuggingFace, OpenAI API), and offers a comprehensive benchmark consisting of 55 datasets. These datasets are divided into 5 key categories that allow for a comprehensive evaluation of model capabilities: factual knowledge, basic skills (e.g., entity extraction and translation), ability to work with long context, effectiveness in RAG (Retrieval-Augmented Generation) tasks, and instruction following accuracy. The main contribution of this work is the creation of an accessible tool that allows research groups to effectively conduct local model evaluation, identify their strengths and weaknesses across the specified categories, and analyze the impact of architectural changes and adaptation processes on the final generation quality.

2 Related Works

The methods for evaluating LLMs consistently lag behind the rapid advancement of the models. Early frameworks, such as `lm-evaluation-harness` (Gao et al., 2024) or HELM (Liang et al., 2023), were originally designed to evaluate foundational models. They initially focused on text continuation tasks and next-token probability estimation. With the industry's shift towards instruction-tuned models (chat models) that interact with users via a message format, these frameworks required significant architectural changes. Although modern versions of these tools have added support for chat templates, their architecture often remains burdened by the legacy of the foundational model era, making it difficult to implement specific evaluation scenarios, such as prefilling the assistant's response (prefill assistant) or fine-grained control over reasoning generation.

For the Russian language, the situation is further severely limited by a shortage of high-quality benchmarks. Historically, the first significant project was Russian SuperGLUE (Shavrina et al., 2020); however, today its tasks are considered solved and insufficiently complex for modern LLMs. The most

relevant and large-scale solution at the moment is the MERA benchmark (Fenogenova et al., 2024). Overall, it is the best existing tool for evaluating Russian-language models now, offering a wide range of tasks. Nevertheless, MERA has several significant drawbacks that limit its applicability for rapid iterative development. First, most of the datasets within the benchmark are already outdated: some are saturated and do not allow for effective differentiation of modern models, while others rather hinder qualitative evaluation. For example, the `ruCodeEval` task (an analogue of `openai_humaneval`) requires the model to simply continue the text, which is characteristic of the pre-instruction era, and the instructions themselves are often out of sync with the evaluation function (e.g., "Given a function description, write a program in Python\n{function}"). Second, evaluation on MERA is based on a fork of the outdated `lm-evaluation-harness` framework, inheriting its architectural limitations. Third, since the answers for the test split are hidden, the evaluation process requires manual uploading of results to the external system (creating a profile on the website, uploading an archive), which makes a fully automated local pipeline impossible.

Regarding evaluation approaches, multiple-choice tasks (e.g., MMLU (Hendrycks et al., 2020)), evaluated based on next-token probability, remain the industry standard for factual knowledge assessment. However, the main problem with such tasks is that they only measure the effect of the pre-training stage, but do not evaluate the generative capabilities of the model and the quality of its alignment (SFT/RL). In practice, a model with a high MMLU score may fall into loops during generation or provide low-quality answers. Accordingly, token probability estimation is insufficient — generative tasks such as translation, summarization, or named entity recognition (NER) are necessary. At the same time, not all generative tasks can be qualitatively evaluated using automatic lexical metrics (e.g., answers to open-ended questions like "Explain to me clearly what quantum physics is"). For such cases, the LLM-as-a-Judge approach (Zheng et al., 2023) is applied, where a strong model acts as a judge. However, this method also has its problems, in particular, susceptibility to biases (e.g., preference for longer answers). To combat these shortcomings, the AlpacaEval work (Dubois et al., 2024) proposed an approach with length control, and later in LLM Arena (Li et al., 2024; Tianle Li*, 2024) - an approach with style control.

Thus, there is a clear need for a unified framework that is originally designed around the message format, allows for fully local and automated evaluation, combines various testing approaches (from probability estimation to LLM-as-a-Judge with style control), and offers an up-to-date set of tasks for the Russian language. LLMTF was developed specifically to meet this need.

3 LLMTF Framework

3.1 Core Framework Design

A key architectural feature of the LLMTF framework is its initial orientation towards the message format as the de facto standard for interacting with modern instruction-tuned models. Unlike outdated solutions where prompts were formed as plain text, LLMTF operates with structured dialogues (roles `system`, `user`, `assistant`). This allows for the most accurate reproduction of real-world model usage conditions.

At the same time, the framework retains full support for foundational models that have not undergone the instruction fine-tuning stage. To evaluate them, a mechanism for creating custom chat templates based on configuration files is implemented. For example, a template can define a structure like `Query: {content}\n\n Response: {content}\n\n`. This approach allows foundational models to be evaluated in the same pipeline as instruction-tuned ones, without violating the message format logic.

LLMTF offers high flexibility in choosing the computational backend. The framework supports local execution via the high-performance `vLLM` library (Kwon et al., 2023) and standard `HuggingFace Transformers` (Wolf et al., 2019), and also allows evaluating models deployed behind an OpenAI-compatible API.

3.2 Evaluation Scenarios

The framework implements three main tasks scenarios for evaluating model quality:

1. **Generative scenario:** The model generates free text in response to a prompt. This scenario is used for translation, summarization, named entity recognition (NER), and open-ended question

answering tasks.

2. **Next-token probability estimation:** The classic approach for multiple-choice tasks (e.g., MMLU). The model estimates the probabilities of given answer options (A, B, C, D) conditioned on the provided context (and special attention was paid to the subtle point that tokens can be whitespace or non-whitespace, which sometimes led to errors in evaluation).
3. **Logsoftmax estimation:** This method is rarely found in standard benchmarks. The prompt and the reference answer are simultaneously fed into the model, after which the `logsoftmax` is calculated only for the answer tokens in a single forward pass. Unlike discrete metrics (option selection) or classic perplexity (PPL), which is too sensitive to any changes, `logsoftmax` estimation allows measuring the model's *confidence* in the correct answer. This is critical when comparing different versions of the same model (with identical tokenization), for example, to evaluate the impact of small changes in hyperparameters or training data composition during adaptation.

3.3 Additional Features

In addition to the main evaluation scenarios, LLMTF implements a number of additional mechanisms that expand its functionality:

- **Dynamic management of few-shot examples:** When evaluating in few-shot mode, the framework automatically controls the length of the final prompt. If adding all examples exceeds the model's token limit (context window), LLMTF autonomously reduces the number of shots until the prompt fits into the context.
- **Prefill Assistant:** A useful feature that allows setting the beginning of the model's response. For example, one can pass the message `{"role": "assistant", "content": "TLDR:\n"}`, and the model will continue generation exactly from this point. This is particularly convenient when it is necessary to reduce the diversity of answers and direct them in a specific direction to solve the target task.
- **Support for Reasoning Models:** With the advent of models utilizing reasoning mechanisms (e.g., o1 or DeepSeek-R1), the need arose to control this process during evaluation. In LLMTF, one can enable reasoning generation and limit its length using the `max_new_tokens_reasoning` parameter. If the model exceeds this limit, the framework forcibly truncates the reasoning, appends the suffix `...the rest of the reasoning chain is hidden due to limit on the length of the thoughts.`, and transitions the model to the final answer generation stage. Essentially, this functionality splits the answering procedure into two stages, but since it is implemented within the model class, the rest of the code processes the final received answer after reasoning, without requiring any modifications to the evaluation logic.
- **Built-in LLM-as-a-Judge:** For tasks where automatic lexical metrics fail (e.g., open-ended questions), LLMTF includes built-in support for evaluation using strong judge models. This can be especially useful in RAG tasks.
- **Logging:** Practice shows that manual analysis of model responses is a key factor in understanding model issues. For this reason, all inputs and outputs when working with any dataset are mandatorily recorded and saved.
- **Batching:** To efficiently utilize the evaluated models (and GPUs), batch evaluation is implemented by default (it is sufficient to pass the corresponding parameter).
- **Extensibility:** The framework's architecture allows for easy addition of new tasks and datasets. To do this, it is sufficient to implement the corresponding class in the `tasks` directory, inheriting it from the base evaluation classes. At the same time, there is no need to account for the presence/absence of reasoning, batching, and other aspects, as they are already implemented in the framework.

4 LLMTF Benchmark

Based on the developed framework, a comprehensive benchmark was assembled to evaluate Russian-language models. The benchmark is provided in three main configurations covering various researcher needs:

1. **Instruct-Full:** The full version for evaluating instruction-tuned models. It includes 55 datasets and involves evaluation in both zero-shot and few-shot modes.
2. **Instruct-Lite:** A lightweight version of `Instruct-Full`. It includes 37 datasets (with reduced number of samples) and runs significantly faster but maintains a high rank correlation with the full version, making it an excellent choice for rapid intermediate evaluation of checkpoints.

4.1 Benchmark Tasks

For convenience of result analysis, the datasets in the instruction configurations are divided into 5 key categories (full table in Appendix B):

- **Knowledge:** Evaluation of the model's factual knowledge, 8 total: ruMMLU, enMMLU(Hendrycks et al., 2020) (both zero-shot and 5-shot), shlepa (movie[2], music[3], law[4], books[5] subsets),
- **Skills:** Evaluation of basic NLP skills. This includes named entity recognition tasks (e.g., NEREL) (Loukachevitch et al., 2021), machine translation (FLORES ru-en/en-ru) (Costa-Jussà et al., 2022), and text classification (e.g., sentiment analysis of Kinopoisk reviews, RuCoLA) (Mikhailov et al., 2022; Loukachevitch et al., 2025), as well as summarization tasks (Gusev, 2020).
- **Long Context:** Evaluation of the model's ability to retain information over large volumes of text (the full version of the LIBRA benchmark) (Churin et al., 2024).
- **RAG (Retrieval-Augmented Generation):** Evaluation of the model's performance with provided context (e.g., SberQuadQA (Efimov et al., 2020). It is important to note an architectural difference: in the `Instruct-Full` version, the quality of answers in RAG tasks is evaluated using LLM-as-a-Judge for maximum accuracy, whereas in `Instruct-Lite`, lexical matching metrics are used to speed up the process.
- **Instruction Following:** Evaluation of how accurately the model follows specified formats and constraints (IFEval, ruIFEval) (Zhou et al., 2023).

For most datasets, either the original dataset prompt was chosen, or a simple and clear prompt was implemented. For example, in the case of the translation task, it looks as follows: "Your task is to translate the text from {input} language to {output} language. Do not provide any explanations or notes and start the translation immediately.\n\n**Text:**\n{input_text}"

Special attention was paid to the named entity recognition (NER) task. Generally, this task is rarely used to evaluate large language models; however, it is a very important task that frequently arises in various settings, from office automation (extracting dates, requisites, etc.) to text anonymization (e.g., to provide commercial data to scientists for research). At the same time, practice shows that specially trained small BERT (Devlin et al., 2019) models still outperform frontier models in a few-shot setup. For these reasons, 4 different datasets were added to the benchmark: collection3 (Mozharova and Loukachevitch, 2016), patient_ner [6], NEREL (Loukachevitch et al., 2021), and NEREL-BIO (Loukachevitch et al., 2023). Furthermore, since the NER task, especially the nested version (NEREL), requires extracting information from the text without changes, and also with repetitions if these repetitions exist in the text, it is not entirely obvious what exact format to demand from language models so that it can be correctly evaluated later. For this reason, 2 different formats were chosen for each of the datasets (totaling 8 tasks): 1. JSON format, where the model is required to return a list `[["type", "entity"], ["type", "entity"], ...]` in the order of entity occurrence in the text; 2. in-place format, where the model must repeat the original text, inserting HTML tags for entity types `"word word ... word <type>entity</type> word ... word"`.

Additionally, the benchmark implements built-in tools for calculating standard deviation using bootstrapping, which allows assessing the statistical significance of the difference between models. Currently, bootstrapping is not applied to RAG tasks in the `Instruct-Full` configuration, as result aggregation requires repeated calls to LLM-as-a-Judge. The output of leaderboards is also implemented both for

[2]https://huggingface.co/datasets/Vikhrmodels/movie_mc
[3]https://huggingface.co/datasets/Vikhrmodels/music_mc
[4]https://huggingface.co/datasets/Vikhrmodels/law_mc
[5]https://huggingface.co/datasets/Vikhrmodels/books_mc
[6]https://huggingface.co/datasets/Mykes/patient_queries_ner

individual tasks and for categories as a whole. Also, the framework logs execution time, allowing the evaluation of computational costs for running a specific task or category.

4.2 LLM-as-a-Judge Arena

A separate, independent component of the benchmark is the arena for evaluating models on general open-ended questions utilizing the LLM-as-a-Judge approach. For this evaluation, we used `Vikhrmodels/ru-arena-general`[7] dataset with 500 open-ended questions. However, the set of input questions can be changed to any other one if necessary when calculating the benchmark.

To construct a more diverse battle grid and improve the robustness of the final metrics, we implemented a multi-baseline strategy. Rather than comparing against a single centralized baseline, each evaluated model engaged in pairwise battles against three distinct anchor models simultaneously: `gpt-4-1106-preview`, `gpt-4o-mini`, and `Qwen3-235B-A22B-Instruct-2507`. `Deepseek V3` was utilized as the independent judge for all pairwise matchups.

4.2.1 Style Control and Bias Mitigation

To mitigate the well-documented problem of judge bias, where an evaluating model unjustifiably prefers a specific structural style or excessively long answers, our framework implements a two-fold mitigation strategy. The first aspect relies on a specialized evaluation prompt, provided in Appendix A. While the final categorical decision of the judge is strictly one-dimensional (returning only `m`, `M`, or `T`), the prompt enforces a structured, internal reasoning process. It instructs the model to prioritize factual accuracy and core substance above all else, explicitly using language fluidity and structural formatting merely as tie-breakers for otherwise identical responses.

The second aspect of our mitigation strategy incorporates a statistical procedure adapted from (Tianle Li*, 2024), designed to disentangle the effect of style from actual substance. Instead of calculating the Elo rating solely based on win rates, we explicitly model stylistic features as independent variables within the Bradley-Terry regression. By treating these specific structural traits similarly to competing models, the regression evaluates them independently, allowing us to calculate a final rating that controls for their direct influence.

5 Experiments and Results

To demonstrate the capabilities of the LLMTF framework and benchmark, large-scale testing of modern local language models was conducted. The sample included both original models from the Qwen family (Yang et al., 2025)(from 4B to 32B parameters) and their specialized adaptations for the Russian language (the Ruadapt (Tihomirov and Chernyshev, 2025; Tikhomirov and Chernyshov, 2024) and T-lite/T-pro families (Stoianov et al., 2025)). Models from other developers, such as YandexGPT-5-Lite-8B, GigaChat3-10B and Vikhr-Nemo-12B-Instruct-R-21-09-24 (Nikolich et al., 2025), were also included in the comparison. All models, except for `Qwen3.5-35B-A3B`, were evaluated on a single graphics card computationally equivalent to an NVIDIA A100. The `Qwen3.5-35B-A3B` model was evaluated on two such cards using tensor parallelism.

5.1 Results on the Instruct-Full Configuration

Table 1 presents the evaluation results of the models on the full benchmark configuration (`Instruct-Full`), which includes 55 datasets. The evaluation of RAG tasks in this configuration was conducted using LLM-as-a-Judge.

An analysis of the results shows a clear dependence of quality on model size: models with 30B+ parameters occupy the leading positions. The benchmark also allows for a clear assessment of the effect of language adaptation. For example, the `T-pro-it-2.1` model (a Qwen adaptation) outperforms the base `Qwen3-32B` in the mean score (0.691 vs. 0.682), showing a significant increase in the factual knowledge category (Knowledge: 0.708 vs. 0.682) and instruction following (IFEval: 0.805 vs. 0.797). At the same time, the `RuadaptQwen3-32B-Instruct` adaptation demonstrates improvements over the base

[7] https://huggingface.co/datasets/Vikhrmodels/ru-arena-general

Table 1: Evaluation results of models on the `Instruct-Full` configuration (with standard deviation calculated via bootstrapping and total execution time in seconds).

Model	Mean	Total Time (s)	IFEval	Knowledge	Long	RAG	Skills
Qwen3.5-27B (2xGPU)	**0.718** ± 0.004	30181	0.815 ± 0.017	0.733 ± 0.006	**0.631** ± 0.010	0.866	**0.503** ± 0.004
Qwen3.5-35B-A3B (2xGPU)	0.701 ± 0.003	12519	**0.823** ± 0.011	**0.742** ± 0.005	0.599 ± 0.006	0.868	0.470 ± 0.004
T-pro-it-2.1	0.691 ± 0.003	66586	0.805 ± 0.013	0.708 ± 0.005	0.620 ± 0.006	0.870	0.454 ± 0.004
Qwen3-32B	0.682 ± 0.003	74544	0.797 ± 0.012	0.682 ± 0.005	0.611 ± 0.005	0.875	0.446 ± 0.004
RuadaptQwen3-32B-Instruct	0.671 ± 0.003	53852	0.751 ± 0.013	0.676 ± 0.005	0.629 ± 0.006	0.872	0.428 ± 0.004
Qwen3-14B	0.670 ± 0.003	32618	0.815 ± 0.011	0.641 ± 0.006	0.603 ± 0.006	0.868	0.422 ± 0.004
Qwen3-30B-A3B-Instruct-2507	0.669 ± 0.003	21103	0.786 ± 0.013	0.676 ± 0.006	0.591 ± 0.005	0.860	0.433 ± 0.004
T-lite-it-2.1	0.657 ± 0.003	15364	0.795 ± 0.013	0.631 ± 0.005	0.607 ± 0.005	0.843	0.410 ± 0.004
T-pro-it-2.0	0.650 ± 0.003	59405	0.672 ± 0.012	0.697 ± 0.005	0.618 ± 0.005	**0.878**	0.386 ± 0.004
Qwen3-8B	0.642 ± 0.003	20459	0.783 ± 0.011	0.596 ± 0.005	0.572 ± 0.005	0.843	0.413 ± 0.004
YandexGPT-5-Lite-8B-instruct	0.633 ± 0.003	23602	0.714 ± 0.013	0.655 ± 0.005	0.540 ± 0.006	0.858	0.396 ± 0.004
Qwen3-4B-Instruct	0.622 ± 0.003	14998	0.783 ± 0.014	0.567 ± 0.005	0.541 ± 0.005	0.828	0.392 ± 0.004
RuadaptQwen3-8B-Hybrid	0.622 ± 0.003	15171	0.746 ± 0.012	0.595 ± 0.005	0.563 ± 0.005	0.834	0.369 ± 0.004
RuadaptQwen3-4B-Instruct	0.613 ± 0.003	11270	0.751 ± 0.013	0.574 ± 0.005	0.565 ± 0.005	0.827	0.347 ± 0.004
Qwen3-4B	0.604 ± 0.003	15003	0.744 ± 0.013	0.551 ± 0.005	0.526 ± 0.006	0.826	0.375 ± 0.004
avibe	0.603 ± 0.004	15251	0.604 ± 0.016	0.632 ± 0.005	0.547 ± 0.005	0.847	0.387 ± 0.004
GigaChat3-10B-A1.8B-bf16	0.595 ± 0.004	12987	0.554 ± 0.018	0.645 ± 0.005	0.567 ± 0.005	0.857	0.353 ± 0.004
T-lite-it-1.0	0.581 ± 0.004	18474	0.579 ± 0.016	0.621 ± 0.005	0.533 ± 0.006	0.844	0.328 ± 0.003
Vikhr-Nemo-12B-Instruct-R-21-09-24	0.569 ± 0.004	25495	0.544 ± 0.016	0.558 ± 0.005	0.535 ± 0.005	0.862	0.344 ± 0.004
RuadaptQwen2.5-7B-Lite-Beta	0.557 ± 0.003	13696	0.514 ± 0.015	0.605 ± 0.005	0.528 ± 0.006	0.841	0.297 ± 0.004

`Qwen3-32B` in working with long context (Long: 0.629) but slightly yields to the original in basic skills (Skills). Additionally, adapted versions provide more generation speed (in terms of sample processing speed, as well as in terms of characters per second). This underscores the importance of comprehensive evaluation across categories, as different adaptation methods (e.g., tokenizer replacement or the specifics of SFT datasets) affect the final capabilities of the model differently.

5.2 Results on the Instruct-Lite Configuration and Correlation Analysis

To test the hypothesis about the possibility of a faster yet reliable model evaluation, the `Instruct-Lite` configuration was used. In it, RAG tasks are evaluated using lexical metrics, and the total number of examples and datasets is reduced.

Table 2: Evaluation results of models on the `Instruct-Lite` configuration (with total execution time in seconds).

Model	Mean	Total Time (s)	IFEval	Knowledge	Long	RAG	Skills
Qwen3.5-27B (2xGPU)	**0.695** ± 0.004	11058	**0.834** ± 0.012	**0.747** ± 0.005	0.635 ± 0.011	0.774 ± 0.008	**0.485** ± 0.007
Qwen3.5-35B-A3B (2xGPU)	0.683 ± 0.004	4624	0.813 ± 0.012	0.743 ± 0.005	0.611 ± 0.011	0.785 ± 0.008	0.461 ± 0.008
T-pro-it-2.1	0.677 ± 0.004	14723	0.810 ± 0.010	0.707 ± 0.005	0.620 ± 0.011	**0.794** ± 0.008	0.453 ± 0.007
Qwen3-30B-A3B-Instruct-2507	0.661 ± 0.004	4991	0.780 ± 0.013	0.674 ± 0.005	**0.659** ± 0.012	0.767 ± 0.008	0.426 ± 0.006
Qwen3-32B	0.661 ± 0.004	19150	0.788 ± 0.013	0.681 ± 0.005	0.633 ± 0.012	0.765 ± 0.008	0.440 ± 0.007
Qwen3-14B	0.648 ± 0.004	8397	0.795 ± 0.011	0.642 ± 0.005	0.612 ± 0.011	0.780 ± 0.007	0.409 ± 0.007
RuadaptQwen3-32B-Instruct	0.647 ± 0.004	14824	0.736 ± 0.013	0.678 ± 0.005	0.640 ± 0.011	0.769 ± 0.008	0.412 ± 0.007
T-lite-it-2.1	0.629 ± 0.004	4133	0.774 ± 0.014	0.632 ± 0.005	0.596 ± 0.011	0.747 ± 0.008	0.394 ± 0.007
T-pro-it-2.0	0.628 ± 0.004	15682	0.655 ± 0.015	0.698 ± 0.005	0.633 ± 0.011	0.771 ± 0.008	0.383 ± 0.007
Qwen3-8B	0.625 ± 0.004	5117	0.768 ± 0.014	0.597 ± 0.005	0.607 ± 0.012	0.746 ± 0.009	0.408 ± 0.007
Qwen3-4B-Instruct	0.605 ± 0.004	3621	0.788 ± 0.012	0.568 ± 0.005	0.573 ± 0.012	0.709 ± 0.009	0.386 ± 0.007
RuadaptQwen3-8B-Hybrid	0.605 ± 0.004	4083	0.740 ± 0.012	0.595 ± 0.005	0.607 ± 0.011	0.722 ± 0.009	0.358 ± 0.006
RuadaptQwen3-4B-Instruct	0.587 ± 0.004	2873	0.736 ± 0.013	0.577 ± 0.005	0.574 ± 0.011	0.703 ± 0.009	0.347 ± 0.007
Qwen3-4B	0.585 ± 0.004	3656	0.754 ± 0.013	0.550 ± 0.005	0.563 ± 0.013	0.696 ± 0.009	0.365 ± 0.007
avibe	0.574 ± 0.005	4057	0.605 ± 0.016	0.631 ± 0.006	0.532 ± 0.011	0.735 ± 0.008	0.369 ± 0.007
GigaChat3-10B-A1.8B-bf16	0.571 ± 0.004	7158	0.548 ± 0.015	0.644 ± 0.005	0.591 ± 0.012	0.742 ± 0.008	0.333 ± 0.006
T-lite-it-1.0	0.562 ± 0.004	4590	0.560 ± 0.015	0.620 ± 0.005	0.562 ± 0.012	0.744 ± 0.008	0.322 ± 0.006
YandexGPT-5-Lite-8B-instruct	0.552 ± 0.004	3396	0.704 ± 0.015	0.445 ± 0.005	0.494 ± 0.011	0.754 ± 0.008	0.364 ± 0.006
Vikhr-Nemo-12B-Instruct-R-21-09-24	0.538 ± 0.004	6528	0.531 ± 0.015	0.559 ± 0.006	0.582 ± 0.012	0.680 ± 0.008	0.339 ± 0.006
RuadaptQwen2.5-7B-Lite-Beta	0.533 ± 0.004	3569	0.502 ± 0.014	0.604 ± 0.005	0.549 ± 0.012	0.718 ± 0.008	0.292 ± 0.006

A comparison of the results in Tables 1 and 2 shows that the `Instruct-Lite` configuration preserves the relative ranking of the models. The Pearson correlation coefficient between the mean scores of the two configurations over 0.95. This proves that for intermediate evaluation of checkpoints during development, the lightweight version of the benchmark can be safely used without losing ranking quality.

5.3 Results on LLM-as-a-Judge Arena

For the calculation of the final Elo rating, `gpt-4o-mini` was selected as the coordinate center, with its baseline rating fixed at exactly 1000. The final battle results, complete with standard deviations, 95% confidence intervals, and median generation lengths (in characters), are detailed in Table 3.

Table 3: Final Elo rankings, standard deviations, 95% confidence intervals, and median generation lengths for models evaluated in the LLM-as-a-Judge Arena.

Model	Elo Rating	STD	95% CI	Median Length (chars)
T-pro-it-2.0	1588	22	[1552, 1635]	2256
Qwen3-235B-A22B-Instruct-2507	1519	9	[1500, 1537]	1858
T-pro-it-2.1	1486	20	[1450, 1531]	1811
Qwen3.5-27B	1474	20	[1438, 1512]	2134
Qwen3.5-35B-A3B	1442	18	[1412, 1479]	2149
Qwen3-30B-A3B-Instruct-2507	1429	19	[1391, 1465]	2152
T-lite-it-2.1	1396	18	[1362, 1429]	1978
Qwen3-32B	1261	14	[1236, 1288]	1551
RuadaptQwen3-32B-Instruct	1235	15	[1209, 1271]	1646
Qwen3-4B-Instruct-2507	1228	15	[1200, 1258]	2267
Qwen3-14B	1201	16	[1168, 1232]	1624
RuadaptQwen3-4B-Instruct	1176	15	[1153, 1207]	1979
avibe	1175	13	[1149, 1198]	1728
RuadaptQwen3-8B-Hybrid	1173	13	[1148, 1200]	1492
Qwen3-8B	1147	16	[1115, 1171]	1729
RuadaptQwen2.5-7B-Lite-Beta	1083	14	[1062, 1113]	1252
gpt-4-1106-preview	1030	9	[1013, 1046]	1261
Qwen3-4B	1001	15	[972, 1028]	1636
T-lite-it-1.0	1000	16	[980, 1038]	1407
gpt-4o-mini	1000	0	[1000, 1000]	876
GigaChat3-10B-A1.8B-bf16	961	15	[931, 987]	959
Vikhr-Nemo-12B-Instruct-R-21-09-24	932	14	[909, 962]	1540
YandexGPT-5-Lite-8B-instruct	794	21	[752, 835]	636

A critical empirical observation during testing involved the persistent influence of verbosity bias exhibited by `Deepseek V3`. Despite actively penalizing style preferences and enforcing hierarchical evaluation frameworks explicitly adopted from established LLM arenas, the correlation between response length and win rate was merely smoothed out rather than completely eliminated.

5.4 Case Study: Automatic Prompt Selection

One of the drawbacks sometimes found in certain benchmarks is that instructions and prompts might have been selected manually based on one or two models, or without any preliminary evaluation of prompt quality at all. For this reason, functionality for automatic prompt selection was introduced into the LLMTF framework, and for all NER tasks in the benchmark, the prompt was selected using the implemented method.

Inspired by AlphaEvolve from Google DeepMind (Novikov et al., 2025) we've built a system capable of automatically optimizing prompts for specific tasks.

5.4.1 Automatic Prompt Selection: Approach

Our system consists out of 6 main components:

- **Solution database** stores evaluated solutions: triplets of prompt template, score and feedback.
- **Sampler** samples existing solution from solution database, in our case, following power law.

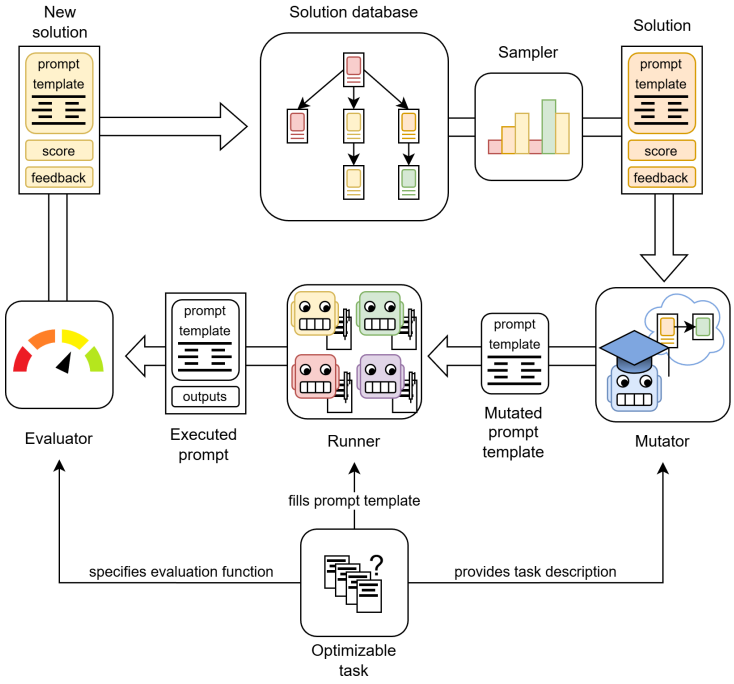

Figure 1: Prompt optimizer architecture

- **Mutator** comes up with new prompt template based on the old one as well as the feedback and task description.
- **Optimizable task** provides task description for mutator and evaluation functions for evaluator. Generates prompts by filling prompt template with information from stored samples.
- **Runner** uses prompts to generate responses from a set of models.
- **Evaluator** scores LLMs' response and creates a feedback, in our case, comparing agents' answers with correct ones for the lowest scored samples.

Algorithm starts by running initial prompt template (provided by the user) and evaluating responses, thus creating a root solution, stored in solution database.

By running a prompt template we mean using it to create messages by filling it with information from train samples and then generating responses on a set of LLMs.

On each iteration sampler feeds random solution from solution database to the mutator, which, based on the task description and feedback on that solution, comes up with new prompt template.

Mutated prompt template is then used by runner to generate responses. Responses are scored by evaluator and feedback is generated. Triplet (prompt template, score and feedback) is our new solution and it is saved in the solution database.

User describes optimizable task, defines set of runner models, mutator model and optimization iteration count as well as few-shot count (0 in our case) number of examples fed to mutator and other parameters.

5.4.2 Automatic Prompt Selection: Results

We tested this system on NER tasks of our framework: Collection3, NEREL, NEREL-BIO, patient queries ner. Each dataset is represented in two version based on required answer format ("json" and "in-place") 8 in total.

As our mutator model we chose `Qwen3-235B-A22B-Instruct-2507`. Our runner models are:

- `ruadapt-qwen3-4b`
- `qwen3-4b-instruct`

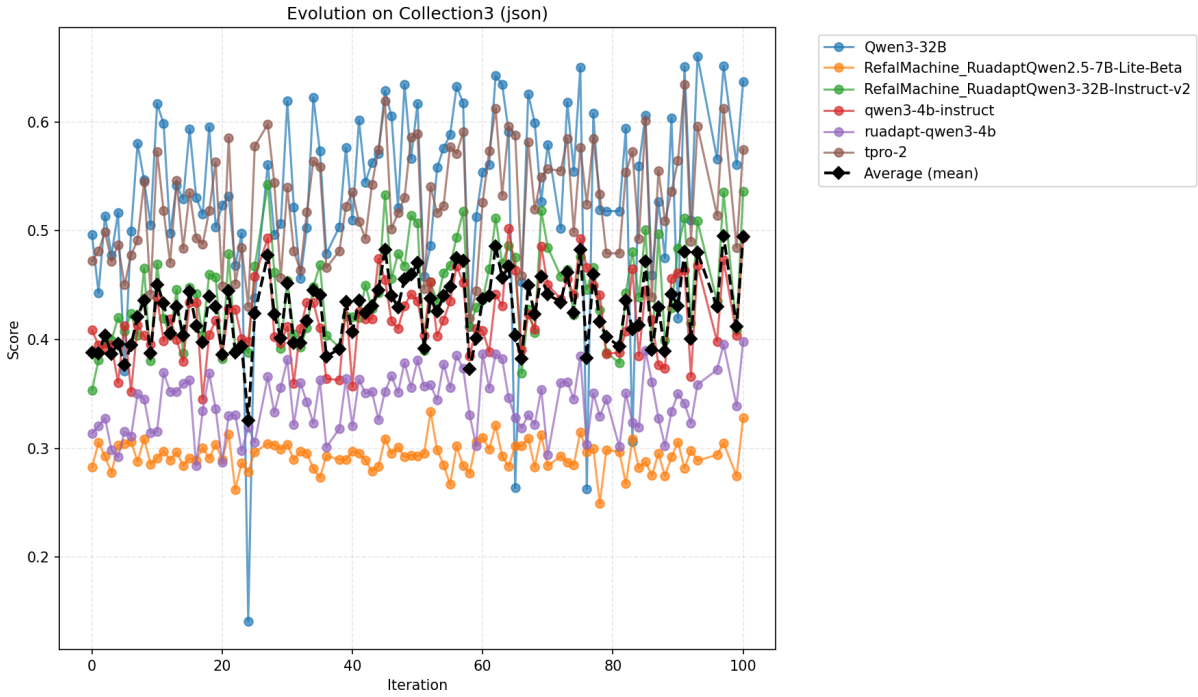

Figure 2: Prompt optimization on Collection3 (json)

- `tpro-2`
- `RefalMachine/RuadaptQwen2.5-7B-Lite-Beta`
- `Qwen3-32B`
- `RefalMachine/RuadaptQwen3-32B-Instruct-v2`

Models hyperparameters:

- temperature: 0.1
- top_k: 40
- top_p: 0.9

For Collection3 we ran our algorithm for 100 iterations on 400 train samples, resulting in higher quality prompts for both “json” and “in-place” variations. For NEREL and NEREL-BIO we settled on 36/40 iterations and 30 train samples due to each sample being much longer compared to other tasks, however we didn’t manage to make substantial changes. For Patient queries ner we used 100 iterations and 200 train samples, in both cases greatly increasing quality of the prompt. Examples of quality plots for training samples can be seen in Tables 2 and 3.

Our results indicate that our algorithm more effective at optimizing simple tasks, yet struggles with complex ones. We believe there are a lot of improvements to be made to this system, for example addition of editable long-term memory, experiences summarization and meta analysis.

5.5 Additional Specialized Benchmarks

To provide a more comprehensive evaluation of the models’ capabilities on complex Russian-language tasks, we integrated two additional benchmarks into our experimental suite. Although these evaluations are executed via independent scripts outside the primary framework logic, they offer unique and highly demanding measurements for long-context processing and retrieval-augmented generation. The aggregated performance metrics for the tested models across both datasets are summarized in Table 4.

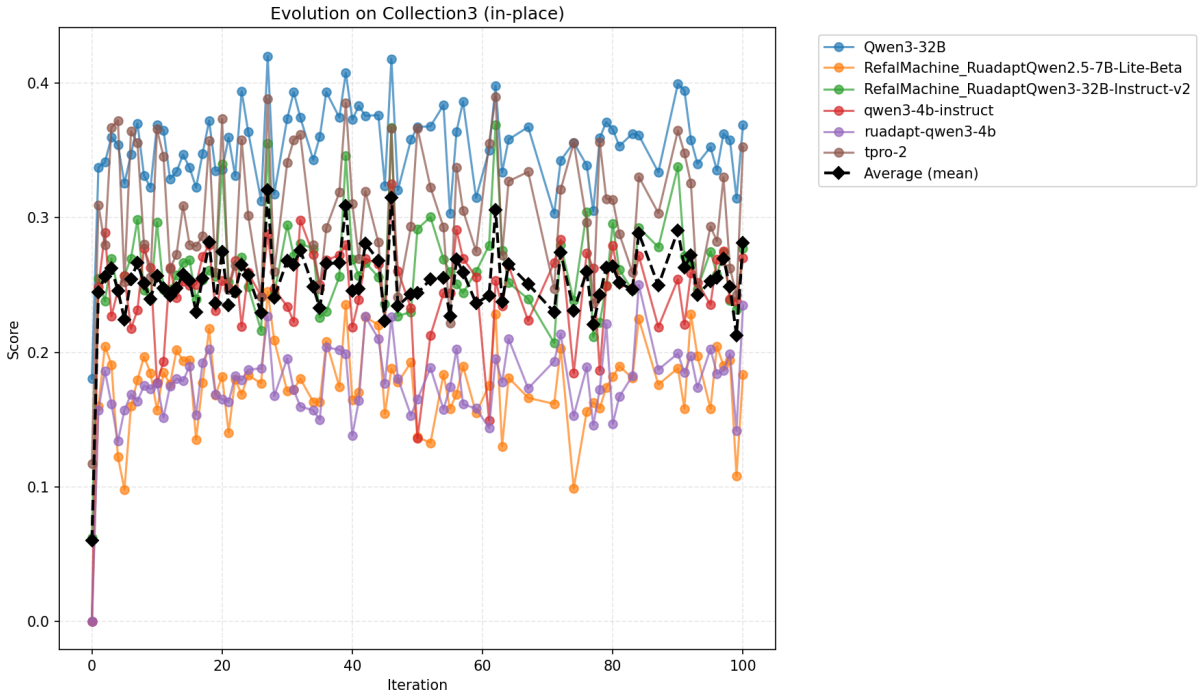

Figure 3: Prompt optimization on Collection3 (in-place)

5.5.1 RuBookSum

The first additional dataset is RuBookSum (Grigoriev et al., 2025), a publicly available benchmark for the abstractive summarization of long-form Russian narrative texts. Conceptually similar to BookSum (Kryściński et al., 2022), this dataset focuses specifically on Russian literature. It was constructed by pairing user-written summaries from the "Narodny Briefly" platform with full literary texts sourced from the Librusec digital library. Following an automated cleaning process to strip HTML tags and user comments, the resulting corpus contains 634 document-summary pairs spanning more than 40 genres. With source texts averaging approximately 35,000 words and target summaries averaging 701 words, the dataset serves as a rigorous test bed for evaluating models on extreme long-context summarization.

5.5.2 RuWikiBench

The second dataset, RuWikiBench (Grigor'ev and Chernyshev, 2025), is a Russian-language benchmark designed to evaluate how well large language models can reproduce Wikipedia-style articles from raw source materials. Built from manually selected articles from the Russian online encyclopedia Ruwiki, each sample pairs an article with the external references explicitly cited in its notes. During the preprocessing phase, the dataset enforces strict grounding constraints. The system retains only content that can be linked to at least one accessible source, removing any unsupported paragraphs and conditionally filtering headings. The remaining reference materials are then split into short snippets, creating a setup that closely mirrors real-world retrieval-augmented generation pipelines. The benchmark is organized around three core tasks: selecting relevant sources, building a hierarchical article outline, and generating the actual article sections. In total, the dataset contains 285 article-level samples comprising 15,686 sources split into 36,860 snippets, with an average outline length of 37 headings. Unlike related benchmarks such as ResearchArena (Kang and Xiong, 2024) (which primarily tests the collection and organization of information), RuWikiBench places a stronger emphasis on the model's ability to synthesize and generate coherent scientific and encyclopedic text.

An analysis of the results presented in Table 4 reveals a stark contrast in task difficulty across the two

Table 4: Performance metrics of the evaluated models on the additional Russian-language benchmarks (RuWikiBench and RuBookSum), sorted by the overall mean score.

Model	Mean	RuWikiBench	RuBookSum
T-pro-it-2.1	56.60	98.29	14.91
Qwen3.5-27B	56.47	98.47	14.47
Qwen3.5-35B-A3B	56.38	97.85	14.90
Qwen3-14B	56.31	97.38	15.23
Qwen3-30B-A3B-Instruct-2507	56.15	97.63	14.66
Vikhr-Nemo-12B-Instruct-R-21-09-24	55.96	97.34	14.57
RuadaptQwen3-8B-Hybrid	55.48	96.43	14.52
RuadaptQwen3-32B-Instruct	55.43	97.07	13.78
T-pro-it-2.0	55.40	98.21	12.59
Qwen3-8B	55.40	96.31	14.48
YandexGPT-5-Lite-8B-instruct	55.26	93.36	17.16
Qwen3-4B	55.20	95.44	14.96
Qwen3-32B	55.17	95.70	14.64
GigaChat3-10B-A1.8B-bf16	54.98	95.15	14.80
RuadaptQwen3-4B-Instruct	54.95	95.70	14.20
T-lite-it-2.1	54.95	95.68	14.21
avibe	54.43	95.94	12.92
Qwen3-4B-Instruct-2507	53.78	93.29	14.26
T-lite-it-1.0	53.66	93.85	13.47
RuadaptQwen2.5-7B-Lite-Beta	52.05	94.00	10.10

domains. Models achieved near-ceiling performance on the RuWikiBench dataset, with scores consistently in the mid-to-high 90s. This indicates that current models possess a strong and reliable capability for retrieval-augmented encyclopedic generation when provided with explicit source snippets. Conversely, the RuBookSum task proved exceptionally challenging, with all models scoring below 18 points, thereby highlighting the persistent difficulty of extreme long-context abstractive summarization. Overall, `T-pro-it-2.1` and large Qwen-based models (such as `Qwen3.5-27B`) demonstrated the best balanced performance and topped the aggregated leaderboard. Notably, `YandexGPT-5-Lite-8B-instruct` exhibited a highly unique performance profile: while it placed near the bottom in RuWikiBench (93.36), it achieved the highest absolute score on the demanding RuBookSum dataset (17.16).

5.6 Framework Performance

An important aspect of LLMTF is its execution speed. Thanks to built-in batching and an optimized architecture, the framework allows for evaluation in a reasonable time. As can be seen from Table 2, a full evaluation of a 7B-8B class model (e.g., `Qwen3-8B` or `T-lite-it-2.1`) on the `Instruct-Lite` configuration takes about 4000-5000 seconds on a single GPU equivalent to an A100. Evaluation on the full `Instruct-Full` configuration for the same models takes about 15000-20000 seconds (4-5.5 hours). At the same time, these evaluations cover 5 different and key task categories for language models, thereby allowing the assessment of quality not just in one or two settings, but across a wide range of tasks. This makes LLMTF a practical tool for daily use in small research laboratories, enabling rapid iteration during model development and adaptation.

6 Conclusion

This paper presents LLMTF, a novel open framework and benchmark for evaluating large language models in Russian. The development and implementation of this tool demonstrate the importance of

creating local, open, and flexible solutions for LLM evaluation. Unlike closed commercial benchmarks aimed at frontier models, LLMTF provides small research groups with the ability to conduct in-depth analysis of developed models within a closed loop. This is particularly relevant for language adaptation tasks, where it is necessary to rapidly iterate and compare hundreds of intermediate checkpoints.

LLMTF solves the problem of evaluation tool fragmentation by offering a unified approach based on the message format, support for various backends, and modern mechanisms (such as prefill assistant and reasoning control). The developed benchmark includes a total of 55 datasets divided into 5 categories and is provided in two main configurations (`Instruct-Full`, `Instruct-Lite`), allowing for flexible customization of the evaluation process for specific research tasks. Furthermore, the framework incorporates an LLM-as-a-Judge Arena with style control to mitigate judge bias, providing a robust evaluation of open-ended generation.

Experimental results confirm that the LLMTF benchmark is capable of effectively ranking models and identifying their strengths and weaknesses. The analysis showed that different methods of adapting Qwen family models to the Russian language lead to uneven quality improvements, underscoring the need for comprehensive evaluation across multiple task categories. An important achievement is the confirmation of a high correlation between the full (`Instruct-Full`) and lightweight (`Instruct-Lite`) benchmark configurations. This allows researchers to significantly save computational resources during the intermediate testing stage. Additionally, the integration of specialized benchmarks like RuBookSum and RuWikiBench highlights the current capabilities and limitations of models in complex Russian-language tasks, such as extreme long-context summarization and retrieval-augmented encyclopedic generation.

In future versions of LLMTF, we plan to expand the set of unique Russian-language datasets, as well as further develop the automatic prompt selection system.

Acknowledgements

The research was supported by the Russian Science Foundation, project Nº 25-11-00191, https://rscf.ru/project/25-11-00191/.

The research was carried out using the MSU-270 supercomputer of Lomonosov Moscow State University.

References

Igor Churin, Murat Apishev, Maria Tikhonova, Denis Shevelev, Aydar Bulatov, Yuri Kuratov, Sergej Averkiev, and Alena Fenogenova. 2024. Long input benchmark for russian analysis. *arXiv preprint arXiv:2408.02439*.

Marta R Costa-Jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, et al. 2022. No language left behind: Scaling human-centered machine translation. *arXiv preprint arXiv:2207.04672*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. // *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, P 4171–4186.

Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B Hashimoto. 2024. Length-controlled alpacaeval: A simple way to debias automatic evaluators. *arXiv preprint arXiv:2404.04475*.

Pavel Efimov, Andrey Chertok, Leonid Boytsov, and Pavel Braslavski. 2020. Sberquad–russian reading comprehension dataset: Description and analysis. // *International Conference of the Cross-Language Evaluation Forum for European Languages*, P 3–15. Springer.

Alena Fenogenova, Artem Chervyakov, Nikita Martynov, Anastasia Kozlova, Maria Tikhonova, Albina Akhmetgareeva, Anton Emelyanov, Denis Shevelev, Pavel Lebedev, Leonid Sinev, et al. 2024. Mera: A comprehensive llm evaluation in russian. // *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, P 9920–9948.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason

Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2024. The language model evaluation harness, 07.

Denis Andreevich Grigor'ev and Daniil Ivanovich Chernyshev. 2025. Ruwikibench: evaluating large language models through replication of encyclopedia articles. *Doklady Rossijskoj Akademii Nauk. Mathematika, Informatika, Processy Upravlenia*, 527:171–181.

Denis A Grigoriev, Daniil V Khudiakov, and Daniil I Chernyshev. 2025. Rubooksum: Dataset for russian literature abstractive summarization. *Supercomputing Frontiers and Innovations*, 12(3):76–89.

Ilya Gusev. 2020. Dataset for automatic summarization of russian news. // *Artificial Intelligence and Natural Language*, P 122–134, Cham. Springer International Publishing.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Hao Kang and Chenyan Xiong. 2024. Researcharena: Benchmarking llms' ability to collect and organize information as research agents.(2024). *URL https://arxiv. org/abs/2406.10291*.

Wojciech Kryściński, Nazneen Rajani, Divyansh Agarwal, Caiming Xiong, and Dragomir Radev. 2022. Booksum: A collection of datasets for long-form narrative summarization. // *Findings of the association for computational linguistics: EMNLP 2022*, P 6536–6558.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. // *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Tianhao Wu, Banghua Zhu, Joseph E Gonzalez, and Ion Stoica. 2024. From crowdsourced data to high-quality benchmarks: Arena-hard and benchbuilder pipeline. *arXiv preprint arXiv:2406.11939*.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, Benjamin Newman, Binhang Yuan, Bobby Yan, Ce Zhang, Christian Alexander Cosgrove, Christopher D Manning, Christopher Re, Diana Acosta-Navas, Drew Arad Hudson, Eric Zelikman, Esin Durmus, Faisal Ladhak, Frieda Rong, Hongyu Ren, Huaxiu Yao, Jue WANG, Keshav Santhanam, Laurel Orr, Lucia Zheng, Mert Yuksekgonul, Mirac Suzgun, Nathan Kim, Neel Guha, Niladri S. Chatterji, Omar Khattab, Peter Henderson, Qian Huang, Ryan Andrew Chi, Sang Michael Xie, Shibani Santurkar, Surya Ganguli, Tatsunori Hashimoto, Thomas Icard, Tianyi Zhang, Vishrav Chaudhary, William Wang, Xuechen Li, Yifan Mai, Yuhui Zhang, and Yuta Koreeda. 2023. Holistic evaluation of language models. *Transactions on Machine Learning Research*. Featured Certification, Expert Certification.

Natalia Loukachevitch, Ekaterina Artemova, Tatiana Batura, Pavel Braslavski, Ilia Denisov, Vladimir Ivanov, Suresh Manandhar, Alexander Pugachev, and Elena Tutubalina. 2021. Nerel: A russian dataset with nested named entities, relations and events.

Natalia Loukachevitch, Suresh Manandhar, Elina Baral, Igor Rozhkov, Pavel Braslavski, Vladimir Ivanov, Tatiana Batura, and Elena Tutubalina. 2023. Nerel-bio: a dataset of biomedical abstracts annotated with nested named entities. *Bioinformatics*, 39(4):btad161.

Natalia Loukachevitch, Lomonosov Moscow, Natalia Tkachenko, Anna Lapanitsyna, Mikhail Tikhomirov, and Nicolay Rusnachenko. 2025. Ruopinionne-2024: Extraction of opinion tuples from russian news texts. // *Proceedings of the International Conference "Dialogue*, volume 2025.

Vladislav Mikhailov, Tatiana Shamardina, Max Ryabinin, Alena Pestova, Ivan Smurov, and Ekaterina Artemova. 2022. Rucola: Russian corpus of linguistic acceptability. *Machine Translation*, 1286:72–8.

Valerie A Mozharova and Natalia V Loukachevitch. 2016. Combining knowledge and crf-based approach to named entity recognition in russian. // *International Conference on Analysis of Images, Social Networks and Texts*, P 185–195. Springer.

Aleksandr Nikolich, Konstantin Korolev, Sergei Bratchikov, Igor Kiselev, and Artem Shelmanov. 2025. Vikhr: The family of open-source instruction-tuned large language models for russian.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco JR Ruiz, Abbas Mehrabian, et al. 2025. Alphaevolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*.

Tatiana Shavrina, Alena Fenogenova, Emelyanov Anton, Denis Shevelev, Ekaterina Artemova, Valentin Malykh, Vladislav Mikhailov, Maria Tikhonova, Andrey Chertok, and Andrey Evlampiev. 2020. Russiansuperglue: A russian language understanding evaluation benchmark. *// Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, P 4717–4726.

Dmitrii Stoianov, Danil Taranets, Olga Tsymboi, Ramil Latypov, Almaz Dautov, Vladislav Kruglikov, Nikita Surkov, German Abramov, Pavel Gein, Dmitry Abulkhanov, Mikhail Gashkov, Viktor Zelenkovskiy, Artem Batalov, Aleksandr Medvedev, and Anatolii Potapov. 2025. T-pro 2.0: An efficient russian hybrid-reasoning model and playground.

Wei-Lin Chiang* Tianle Li*, Anastasios Angelopoulos*. 2024. Does style matter? disentangling style and substance in chatbot arena, August.

Mikhail Mikhailovich Tihomirov and Daniil Ivanovich Chernyshev. 2025. Ruadapt: cost-effective large language model lingual adaptation. *Doklady Rossijskoj Akademii Nauk. Mathematika, Informatika, Processy Upravlenia*, 527:290–300.

Mikhail Tikhomirov and Daniil Chernyshov. 2024. Facilitating large language model russian adaptation with learned embedding propagation. *Journal of Language and Education*, 10(4):130–145.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*.

A LLM-as-a-Judge Evaluation Prompt

The following prompt was used for the `Deepseek V3` judge to evaluate and compare the responses of two models in a pairwise battle. The prompt explicitly instructs the judge to prioritize core factual accuracy and helpfulness over stylistic considerations.

I will provide you with a prompt and two responses from models 'm' and 'M'. Your task is to determine which model provided a better response by following a strict evaluation procedure. The primary language of the content is Russian.

Prompt:
{instruction}

Model 'm' Response:
"""{output_m}"""

Model 'M' Response:
"""{output_M}"""

Evaluation Procedure
To make your decision, you must evaluate the responses sequentially according to three criteria, in order of priority:

Step 1: Core Answer Quality (Highest Priority - Weight 3)

- **Relevance and Completeness**: How accurately and thoroughly does the response address the user's prompt?
- **Factual Accuracy and Helpfulness**: Are the provided facts correct? Does the response effectively solve the user's task or answer their question?
- **IMPORTANT**: At this stage, you **must ignore** all aspects of grammar, spelling, style, and formatting. Focus solely on the substance and content of the answer.

Step 2: Language Quality (Secondary Priority - Weight 2)

- **Grammar and Spelling**: Is the text free of spelling, punctuation, or grammatical errors?
- **Fluency and Naturalness**: Does the text read smoothly and naturally in Russian? Is the word choice appropriate?

Step 3: Style and Structure (Lowest Priority - Weight 1)

- **Readability and Formatting**: Is the text easy to read? Does it use paragraphs, lists, or bolding to improve clarity?
- **Clarity of Exposition**: Is the thought process presented in a clear and logical manner?

Decision-Making Rules

Use a hierarchical approach based on your evaluation:

1. **First, compare the responses based on 'Core Answer Quality' (Step 1)**. If one response is significantly better in substance, relevance, and accuracy, choose it immediately, even if it has flaws in language or style.
2. **If the responses are roughly equal in 'Core Answer Quality'**, use **'Language Quality' (Step 2)** as the tie-breaker. Choose the response that is more grammatically correct and fluent.
3. **If the responses are equal in both 'Core Answer Quality' and 'Language Quality'**, use **'Style and Structure' (Step 3)** as the final deciding factor. Choose the response that is better structured and clearer.
4. **If the responses remain tied after all three steps**, you must consider it a tie.

Your Response

Which response is better? Respond with **only a single character** and no other text or explanation.

- If model 'm' is clearly better according to the rules above, respond with: `m`
- If model 'M' is clearly better according to the rules above, respond with: `M`
- If the responses are of equivalent quality or the choice is a toss-up, respond with: `T`

B LLMTF Datasets

Table 5: Benchmark datasets and configurations

#	Category	Task ID	Source Dataset (HF)	Variant	Metric	Few-shot	Test Samples
1	Knowledge	nlpcoreteam/ruMMLU_zero_shot	NLPCoreTeam/mmlu_ru	Zero-shot	Accuracy	0	14,042
2		nlpcoreteam/enMMLU_zero_shot	NLPCoreTeam/mmlu_ru	Zero-shot	Accuracy	0	14,042
3		shlepa/movie_mc	Vikhrmodels/movie_mc	—	Accuracy	0	498
4		shlepa/music_mc	Vikhrmodels/music_mc	—	Accuracy	0	493
5		shlepa/law_mc	Vikhrmodels/law_mc	—	Accuracy	0	1,000
6		shlepa/books_mc	Vikhrmodels/books_mc	—	Accuracy	0	500
7		nlpcoreteam/ruMMLU_few_shot	NLPCoreTeam/mmlu_ru	Few-shot (5)	Accuracy	5	14,042
8		nlpcoreteam/enMMLU_few_shot	NLPCoreTeam/mmlu_ru	Few-shot (5)	Accuracy	5	14,042
9	Skills	darumeru/flores_ru_en	RefalMachine/darumeru (flores)	ru → en	ROUGE-L	3	987
10		darumeru/flores_en_ru	RefalMachine/darumeru (flores)	en → ru	ROUGE-L	3	987
11		daru/treewayabstractive	dichspace/daru_treeway_eval	—	ROUGE-L	0	2,619
12		ilyagusev/gazeta	IlyaGusev/gazeta	—	ROUGE-L	3	6,793
13		ai-forever/kinopoisk	ai-forever/kinopoisk-sentiment-classification	—	F1-macro	5	1,500
14		ruopinionne	RefalMachine/ruopinionne	Full (S, T, E, P)	Tuple-F1	5	2,556
15		ruopinionnesimple	RefalMachine/ruopinionne	Simple (S, T, P)	Tuple-F1	5	2,556
16		MalakhovIlya/NEREL-(json)_few_shot	MalakhovIlya/NEREL	JSON output	F1-macro	3	93 / 885
17		nerel-ds/NEREL-BIO-(json)_few_shot	RefalMachine/nerel-bio-simple	JSON output	F1-macro	3	77 / 600
18		Mykes/patient_queries_ner-(json)_few_shot	Mykes/patient_queries_ner	JSON output	F1-macro	3	239
19		MalakhovIlya/NEREL-(in-place)_few_shot	MalakhovIlya/NEREL	In-place markup	F1-macro	3	93 / 885
20		nerel-ds/NEREL-BIO-(in-place)_few_shot	RefalMachine/nerel-bio-simple	In-place markup	F1-macro	3	77 / 600
21		Mykes/patient_queries_ner-(in-place)_few_shot	Mykes/patient_queries_ner	In-place markup	F1-macro	3	239
22		russiannlp/rucola_custom	RussianNLP/rucola	—	MCC	5	2,787
23		RCC-MSU/collection3-(in-place)_few_shot	RCC-MSU/collection3	In-place markup	F1-macro	3	1,922
24		RCC-MSU/collection3-(json)_few_shot	RCC-MSU/collection3	JSON output	F1-macro	3	1,922
25	RAG	bearberry/rubqqa	bearberry/rubqqa	Task-first prompt	ROUGE-L, LLM-Judge	0	2,910
26		bearberry/rubqqa_data_first	bearberry/rubqqa	Data-first prompt	ROUGE-L, LLM-Judge	0	2,910
27		bearberry/rus_tydiqa	bearberry/rus_tydiqa	Task-first prompt	ROUGE-L, LLM-Judge	0	6,490
28		bearberry/rus_tydiqa_data_first	bearberry/rus_tydiqa	Data-first prompt	ROUGE-L, LLM-Judge	0	6,490
29		bearberry/sberquadqa	bearberry/sberquadqa	Task-first prompt	ROUGE-L, LLM-Judge	0	50,364
30		bearberry/sberquadqa_data_first	bearberry/sberquadqa	Data-first prompt	ROUGE-L, LLM-Judge	0	50,364
31		bearberry/rus_xquadqa	bearberry/rus_xquadqa	Task-first prompt	ROUGE-L, LLM-Judge	0	1,190
32		bearberry/rus_xquadqa_data_first	bearberry/rus_xquadqa	Data-first prompt	ROUGE-L, LLM-Judge	0	1,190
33	IFEval	ifeval/ruIFEval	Local (ruIFEval_v0.1.jsonl)	—	Prompt-level Accuracy	0	541
34		ifeval/enIFEval	Local (enIFEval.jsonl)	—	Prompt-level Accuracy	0	541
35	Long	forever/libra/passkey	ai-forever/LIBRA_old	passkey	EM	0	1,200
36		forever/libra/matreshka_yes_no	ai-forever/LIBRA_old	matreshka_yes_no	EM	0	1,799
37		forever/libra/matreshka_names	ai-forever/LIBRA_old	matreshka_names	EM	0	900
38		forever/libra/passkey_with_librusec	ai-forever/LIBRA_old	passkey_with_librusec	EM	0	1,200
39		forever/libra/librusec_history	ai-forever/LIBRA_old	librusec_history	EM	0	128
40		forever/libra/ru_gsm100	ai-forever/LIBRA_old	ru_gsm100	EM	0	100
41		forever/libra/ru_sci_passage_count	ai-forever/LIBRA_old	ru_sci_passage_count	Count-score	0	600
42		forever/libra/ru_2wikimultihopqa	ai-forever/LIBRA_old	ru_2wikimultihopqa	EM	0	300
43		forever/libra/long_context_multiq	ai-forever/LIBRA_old	long_context_multiq	EM	0	1,200
44		forever/libra/ru_sci_abstract_retrieval	ai-forever/LIBRA_old	ru_sci_abstract_retrieval	EM	0	1,240
45		forever/libra/ru_trec	ai-forever/LIBRA_old	ru_trec	EM	0	300
46		forever/libra/ru_sci_fi	ai-forever/LIBRA_old	ru_sci_fi	EM	0	64
47		forever/libra/librusec_mhqa	ai-forever/LIBRA_old	librusec_mhqa	EM	0	384
48		forever/libra/ru_babilong_qa1	ai-forever/LIBRA_old	ru_babilong_qa1	EM	0	600
49		forever/libra/ru_babilong_qa2	ai-forever/LIBRA_old	ru_babilong_qa2	EM	0	600
50		forever/libra/ru_babilong_qa3	ai-forever/LIBRA_old	ru_babilong_qa3	EM	0	600
51		forever/libra/ru_babilong_qa4	ai-forever/LIBRA_old	ru_babilong_qa4	EM	0	600
52		forever/libra/ru_babilong_qa5	ai-forever/LIBRA_old	ru_babilong_qa5	EM	0	600
53		forever/libra/ru_quality	ai-forever/LIBRA_old	ru_quality	EM	0	202
54		forever/libra/ru_tpo	ai-forever/LIBRA_old	ru_tpo	EM	0	251
55		forever/libra/ru_qasper	ai-forever/LIBRA_old	ru_qasper	F1	0	203