Evalverse: Unified and Accessible Library for Large Language Model Evaluation

Jihoo Kim, Wonho Song, Dahyun Kim, Yunsu Kim, Yungi Kim, Chanjun Park[†]

Upstage AI
{jerry, ynot, kdahyun, yoonsoo, eddie, chanjun.park}@upstage.ai

Abstract

This paper introduces Evalverse[1], a novel library that streamlines the evaluation of Large Language Models (LLMs) by unifying disparate evaluation tools into a single, user-friendly framework. Evalverse enables individuals with limited knowledge of artificial intelligence to easily request LLM evaluations and receive detailed reports, facilitated by an integration with communication platforms like Slack. Thus, Evalverse serves as a powerful tool for the comprehensive assessment of LLMs, offering both researchers and practitioners a centralized and easily accessible evaluation framework. Finally, we also provide a demo video for Evalverse, showcasing its capabilities and implementation in a two-minute format[2].

1 Introduction

In recent years, the rapid advancements in Large Language Models (LLMs) have significantly transformed the computational linguistics landscape, presenting novel opportunities and challenges (Wei et al., 2022; Zhao et al., 2023). Due to the vast scale and complexity of LLMs (Kaplan et al., 2020), they have demonstrated remarkable capabilities across numerous applications (Hadi et al., 2023), ranging from natural language understanding and generation to more specialized tasks such as summarization (Jin et al., 2024), translation (Hendy et al., 2023), and question-answering (Zhuang et al., 2024). However, the sheer pace of LLM development has led to a fragmented ecosystem of evaluation tools and methodologies (Chang et al., 2023; Guo et al., 2023). This fragmentation not only hinders the comparative assessment of LLMs, but also places a considerable barrier to entry for both researchers and practitioners.

Recognizing the critical need for a more unified and accessible framework for LLM evaluation, we introduce Evalverse with the overview depicted in Figure 1 – a novel library that centralizes various evaluation methodologies. Evalverse built such that it can function as a unified and expandable library for LLM evaluation while also lowering the technical barrier to entry of LLM evaluation.

To achieve the former, we integrate existing evaluation frameworks, such as lm-evaluation-harness (Gao et al., 2023) and FastChat (Zheng et al., 2024), as submodules, allowing an easy extension of new benchmarks. These added submodules can reflect recent changes, allowing Evalverse to remain up-to-date with relative ease. On the other hand, we also implement no-code evaluation features that utilize communication platforms such as Slack[3], making LLM evaluation more accesible for individuals with less programming proficiency.

This paper provides an in-depth examination of the architecture and functionality of Evalverse, illustrating how it addresses the current challenges in LLM evaluation. Some of the key features of Evalverse include no-code evaluation and a unified and expandable library for LLM benchmarks, enhancing the efficiency and accessibility.

2 Related Work and Background

2.1 LLM Evaluation Aspects

There are multiple aspects of LLM evaluation, which can be divided into the following four categories: i) general performance; ii) performance for chat applications; iii) performance for Retrieval Augmented Generation (RAG) (Lewis et al., 2020); iv) performance for various domains.

General performance. The Hugging Face Open LLM Leaderboard (Beeching et al., 2023) is primarily utilized for evaluation general performance. The leaderboard uses a total of six benchmarks,

[†] Corresponding Author

[1] https://github.com/UpstageAI/evalverse

[2] https://www.youtube.com/watch?v=-VviAutjpgM

[3] https://slack.com/

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing System Demonstrations, pages 25–33
November 12-16, 2024 ©2024 Association for Computational Linguistics

	General	Chat			RAG	Domain			Additional Features		
Evaluation Framework	H6 Avg	MT-Bench	IFEval	EQ-Bench	RGB	Finance	Medical	Law	Leaderbaord	Eval Report	No-Code Eval
lm-evaluation-harness	O	✗	O	O	✗	✗	O	✗	✗	✗	✗
FastChat	✗	O	✗	✗	✗	✗	✗	✗	O	✗	✗
OpenCompass	O	O	O	✗	✗	O	O	O	O	✗	O
LightEval	O	✗	O	✗	✗	✗	O	O	O	✗	✗
Evalverse (Ours)	O	O	O	O	△	△	△	△	✗	O	O

Table 1: Comparison between LLM evaluation frameworks. Note that Evalverse incorporates all of the shown benchmarks in for "General" and "Chat" evaluation, respectively. Further, we are actively expanding Evalverse to include benchmarks for RAG and other domain specific evaluations as well, indicated by the blue triangle. Further, Evalverse supports no-code evaluation and reports, unlike other frameworks.

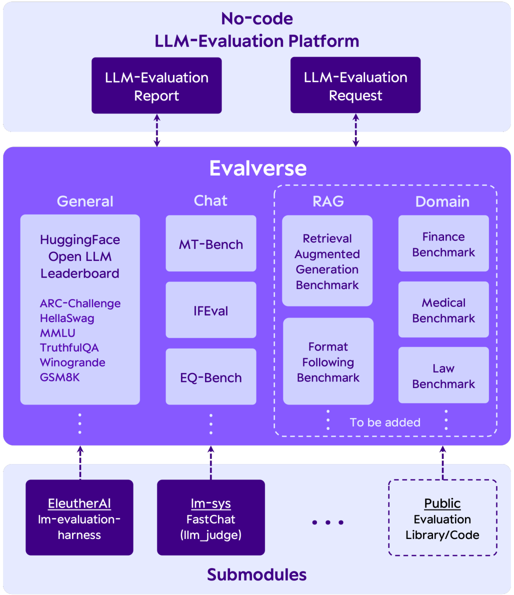


Figure 1: Overview of Evalverse. Users can interact with Evalverse in a no-code manner. External benchmark frameworks are integrated as submodules.

AI2 Reasoning Challenge (Clark et al., 2018), HellaSwag (Zellers et al., 2019), Massive Multitask Language Understanding (MMLU) (Hendrycks et al., 2020), TruthfulQA (Lin et al., 2021), Winogrande (Sakaguchi et al., 2021), and GSM8k (Cobbe et al., 2021), and the average of these scores is commonly referred to as H6 Avg.

Performance for chat applications. One of the primary use cases for LLMs is chat applications (Team et al., 2023; Achiam et al., 2023). It is crucial to measure whether LLMs follow the user's instructions properly and work effectively in a multi-turn environment. The representative methods for evaluating these chat abilities are MT-Bench (Zheng et al., 2024), IFEval (Zhou et al., 2023), and EQ-Bench (Paech, 2023).

Performance for RAG. Pre-trained or fine-tuned LLMs alone may not be sufficient to meet business-level requirements. Therefore, RAG can be an appropriate solution, which involves retrieving documents related to the user queries and providing them as input context to the LLMs. To judge the performance of the LLMs in terms of RAG performance, Chen et al. (2023) introduces Retrieval-Augmented Generation Benchmark (RGB). Further, Xia et al. (2024) presents Format-Following benchmark (FoFo) for evaluating the ability to follow specific formats, which is important for more complex RAG applications as they heavily depend on the intermediate outputs adhering to pre-defined structures.

Performance for various domains. There are many applications of LLMs in various domains such as finance, healthcare, and law. The FinGPT Benchmark (Wang et al., 2023), MultiMedQA (Singhal et al., 2023), and LegalBench (Guha et al., 2022) correspond to the financial, medical, and legal domain, respectively.

2.2 LLM Evaluation Frameworks

There exists other evaluation frameworks for measuring the performance of LLMs across multiple benchmarks. Eleuther AI's lm-evaluation-harness (Gao et al., 2023) is a widely used framework, where over 60 tasks are supported such as H6 Avg, IFEval, and EQ-Bench. LMSYS Org's FastChat (Zheng et al., 2024) supports LLM-Judge to evaluate MT-Bench. OpenCompass[4] is an LLM evaluation platform supporting evaluations not only for H6 Avg, MT-Bench and IFEval but also for multiple domains like Finance, Healthcare, and Law. The most recently released LightEval[5] by HuggingFace is built on top of EleutherAI's lm-evaluation harness. The difference between these frameworks

[4] https://github.com/open-compass/OpenCompass/

[5] https://github.com/huggingface/lighteval

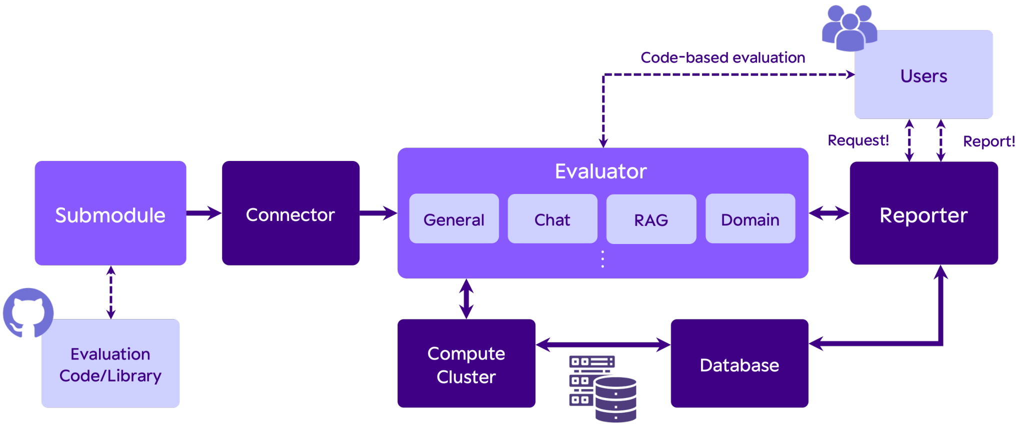


Figure 2: The system architecture of Evalverse. Users can use the Evaluator directly for code-based evaluation, or interact with the Reporter for a no-code approach to LLM evaluation.

and Evalverse is summarized in Table 1.

3 Evalverse

3.1 Why Evalverse?

The core motivation behind Evalverse is to facilitate a unified and expandable library for LLM evaluation, while also being more easily accessible than other existing frameworks. To that end, we integrate benchmarks in a way that is less burdensome to keep them up-to-date. Further, we engineer a *no-code approach* for LLM evaluation, thereby broadening the user base beyond those with coding proficiency. This sets Evalverse apart from conventional evaluation frameworks (Resnik and Lin, 2010) that often necessitate programming skills.

This paper elucidates the architecture and functional capabilities. We posit that the design principles adopted in Evalverse could serve as a blueprint for other evaluation frameworks as well.

3.2 Evalverse Architecture

We explain the system architecture of Evalverse to facilitate a unified evaluation framework whilst also supporting no-code evaluation. Evalverse consists of the following six primary components: Submodule, Connector, Evaluator, Compute Cluster, Database, and Reporter. The overall architecture of Evalverse is illustrated in Figure 2.

Submodule. The Submodule serves as the evaluation engine that is responsible for the heavy lifting involved in evaluating LLMs. Publicly available LLM evaluation libraries can be integrated into Evalverse as submodules. This component makes Evalverse expandable, thereby ensuring that the library remains up-to-date.

Connector. The Connector plays a role in linking the Submodules with the Evaluator. It contains evaluation scripts, along with the necessary arguments, from various external libraries.

Evaluator. The Evaluator performs the requested evaluations on the Compute Cluster by utilizing the evaluation scripts from the Connector. The Evaluator can receive evaluation requests either from the Reporter, which facilitates a no-code evaluation approach, or directly from the end-user for code-based evaluation.

Compute Cluster. The Compute Cluster is the collection of hardware accelerators needed to execute the LLM evaluation processes. When the Evaluator schedules an evaluation job to be ran, the Compute Cluster fetches the required model and data files from the Database. The results of the evaluation jobs are sent to the Database for storage.

Database. The Database stores the model files and data needed in the evaluation processes, along with evaluation results. The stored evaluation results are used by the Reporter to create evaluation reports for the user.

Reporter. The Reporter handles the evaluation and report requests sent by the users, allowing for a no-code approach to LLM evaluation. The Reporter sends the requested evaluation jobs to the Evaluator and fetches the evaluation results from the Database, which are sent to the user via an external communication platform such as Slack. Through this, users can receive table and figure that summarize evaluation results.

3.3 Evalverse Functionality

We detail the no-code, unified, and expandable evaluation as core functionalities of Evalverse, derived from its system architecture.

No-code evaluation. Evalverse supports no-code evaluation using the Reporter explained in the previous section. We have chosen Slack as the initial external communication tool for the no-code evaluation feature, owing to its popular use among numerous companies and communities alike.[6] A detailed example usage of no-code evaluation is given in Section 3.4.

Further, Evalverse also supports a no-code evaluation report feature, where average scores and rankings for just the selected models are retrieved from the Database. This functionality allows non-technical personnel to proactively retrieve evaluation results without having to ask someone with more programming proficiency. Example usage is illustrated in Section 3.4.

Unified and expandable evaluation. For unified and expandable evaluation, Evalverse utilizes Git submodules[7] to integrate external evaluation frameworks such as lm-evaluation-harness (Gao et al., 2023) and FastChat (Zheng et al., 2024). Thus, one can easily add new submodules to support more external evaluation frameworks. Not only that, one can always fetch upstream changes of the submodules to stay up-to-date with evaluation processes in the fast-paced LLM field.

Evalverse includes IFEval (Zhou et al., 2023) and EQ-Bench (Paech, 2023) which are designed for more nuanced evaluation of LLMs for chat applications. Furthermore, RGB (Chen et al., 2023), FoFo (Xia et al., 2024), FinGPT (Wang et al., 2023), MultiMedQA (Liu et al., 2024) and LegalBench (Guha et al., 2022) are being added to expand the evaluation suite to RAG, finance, medical, and legal capabilities, respectively.

The unified nature of Evalverse allows a one-step installation of all the required dependencies for different LLM evaluations. Further, one can aggregate and manage common arguments across multiple benchmarks, such as model name or path.

3.4 Evalverse Tour

We demonstrate how to use Evalverse from installation to executing no-code and code-based evaluation processes.

Installation. One can clone the Evalverse repository and install all the necessary packages at once with the following command:

```
# Evalverse and submodules
git clone --recursive https://github.com
    /UpstageAI/evalverse

# Install the required packages
cd evalverse
pip install -e .
```

Unlike a typical `git clone`, the additional `--recursive` option ensures that the submodules are also cloned.

Configuration. We recommend using a ".env" file to configure the required environment variables (*e.g.*, API keys), similar to the following example:

```
# .env
OPENAI_API_KEY=sk-...

SLACK_BOT_TOKEN=xoxb-...
SLACK_APP_TOKEN=xapp-...
```

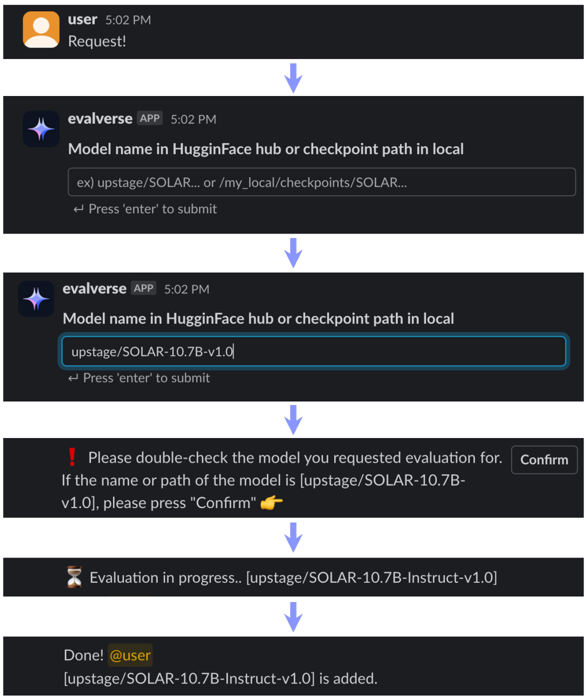
The "OpenAI_API_Key" is used to call the GPT-4 API (OpenAI, 2023) in LLM-as-judge evaluation methods such as the MT-bench implemented in FastChat (Zheng et al., 2024). The "Slack_BOT_Token" and "Slack_APP_Token" are needed for the no-code evaluation feature via Slack, implemented in the Reporter.

No-code evaluation. Evalverse supports no-code evaluation via Slack requests, as depicted in Figure 3. The user types "Request!" in a direct message or Slack channel with an activate Evalverse Slack bot. The Slack bot asks the user to enter the model name in the Huggingface hub (Wolf et al., 2019) or the local model directory path. Then, the Slack bot asks the user for confirmation and then launches an evaluation job on the remote Compute Cluster. The Compute Cluster fetches the model file and necessary benchmark data caches (if present) from the Database and executes the evaluation process. After the evaluation job is finished, an indication is sent to the user. During the entire process, the user only interacts with the Slack bot with no programming involved.

No-code evaluation results look-up. In addition to requesting new evaluations, Evalverse can also provide evaluation reports on finished evaluation in

[6] Expansion to other communication tools are set as important milestones in the development road-map.

[7] https://git-scm.com/book/en/v2/Git-Tools-Submodules



user 5:02 PM
Request!

evalverse APP 5:02 PM
Model name in HugginFace hub or checkpoint path in local
ex) upstage/SOLAR... or /my_local/checkpoints/SOLAR...
↵ Press 'enter' to submit

evalverse APP 5:02 PM
Model name in HugginFace hub or checkpoint path in local
upstage/SOLAR-10.7B-v1.0
↵ Press 'enter' to submit

❗ Please double-check the model you requested evaluation for. If the name or path of the model is [upstage/SOLAR-10.7B-v1.0], please press "Confirm" 👉

⏳ Evaluation in progress.. [upstage/SOLAR-10.7B-Instruct-v1.0]

Done! @user
[upstage/SOLAR-10.7B-Instruct-v1.0] is added.

Figure 3: No-code evaluation request with Slack bot.

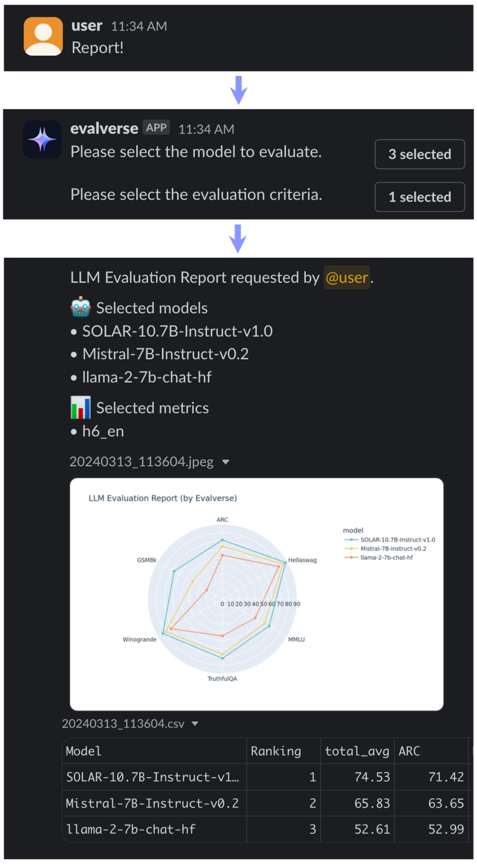
Engine	# Few-shots	Dtype	SOLAR-10.7B-v1.0	Mistral-7B-v0.1
hf	5	float16	64.38	62.59
vllm	5	float16	64.36	62.65
hf	1	float16	62.54	60.56
hf	5	int8	64.24	62.51

Table 2: MMLU scores depending on different inference engine options such as “hf”, HuggingFace transformers (Jain, 2022), or “vllm”, the vLLM framework (Kwon et al., 2023), and other options such as the data type (“dtype”) and number of few-shots.

a no-code manner. To receive the evaluation report, the user first types “Report!”, similar to the evaluation request. Then, the Slack bot will ask the user to select the models and evaluation criteria. For the selected model and evaluation criteria, Evalverse calculates the average scores and rankings using the evaluation results stored in the Database and provides a report with a performance table and a visualized graph as illustrated in Figure 4.

Code-based evaluation. In addition to the no-code evaluation features, one can conduct code-based evaluations for a more fine-grained control. Evalverse supports running multiple benchmarks with a single Python script as detailed below.

```
python3 evaluator.py \
    --ckpt_path {model_path} \
    --{benchmark_A} \
    --{benchmark_B} \
    --{args}
```



user 11:34 AM
Report!

evalverse APP 11:34 AM
Please select the model to evaluate. 3 selected

Please select the evaluation criteria. 1 selected

LLM Evaluation Report requested by @user.

🤖 Selected models
- SOLAR-10.7B-Instruct-v1.0
- Mistral-7B-Instruct-v0.2
- llama-2-7b-chat-hf

📊 Selected metrics
- h6_en

20240313_113604.jpeg ▾

20240313_113604.csv ▾

Model	Ranking	total_avg	ARC
SOLAR-10.7B-Instruct-v1…	1	74.53	71.42
Mistral-7B-Instruct-v0.2	2	65.83	63.65
llama-2-7b-chat-hf	3	52.61	52.99

Figure 4: No-code evaluation report with Slack bot.

The `--ckpt-path` is a common argument used in all benchmarks, where the model name from the Hugging Face Hub or the local path of the model is given. To evaluate a specific set of benchmarks, one can do so by adding the corresponding argument. For a concrete example, the `--h6_en` argument is used for the H6 benchmark in the Open LLM Leaderboard (Beeching et al., 2023) implemented with lm-evaluation-harness, and the `--mt_bench` argument is used for MT-Bench implemented with FastChat. Then, using 8 GPUs for data parallelism, one can perform evaluation on the aforementioned two benchmarks with the following command:

```
python3 evaluator.py \
    --ckpt_path upstage/SOLAR-10.7B-Instruct-v1.0 \
    --h6_en \
    --mt_bench \
    --data_parallel 8
```

4 Evaluation Comparisons

We compare the evaluation results using Evalverse and the original implementation whenever possible. The evaluated models include various open-source models such as Llama 2 (Touvron et al., 2023),

Model	H6		MT-Bench		EQ-Bench		IFEval	
	orig	evalverse	orig	evalverse	orig	evalverse	orig	evalverse
SOLAR 10.7B Instruct	74.53	74.53	7.569	7.580	72.31	73.34	-	0.5370
Mistral 7B Instruct	65.82	65.82	7.466	7.600	70.05	66.57	-	0.5823
Llama 2 7B Chat	52.61	52.61	6.541	6.509	35.09	37.76	-	0.4325
Qwen 1.5 7B Chat	55.66	55.66	7.606	7.575	57.33	51.33	-	0.4797

Table 3: Comparison of evaluation results between the original (orig) repository and Evalverse for H6, MT-Bench, and EQ-Bench. The results show small differences compared to the original for benchmarks with no intentional modifications (H6, MT-Bench). The difference in EQ-Bench is mostly due to an intended modification of the prompts used in evaluation.

Tools	Evaluation Time			
	H6	MT-Bench	EQ-Bench	IFEval
Original repo	32.3	7.6	11.2	-
Evalverse	31.2	7.5	5.6	2.45

Table 4: Evaluation time differences between the original repository and Evalverse for the Solar 10.7B Instruct model. Time units are expressed in minutes.

Mistral (Jiang et al., 2023), Qwen 1.5 (Bai et al., 2023), and SOLAR (Kim et al., 2023).

Differences from the original implementation. When creating Evalverse, we adopted external frameworks as submodules, sometimes with intentional modifications. First, the EQ-Bench in Evalverse uses the prompt in the original release of EQ-Bench version 2, whereas the upstream original repository uses the prompt in version 2.2. Version 2 uses revision prompts where it asks the model to revise its own answers if needed. In contrast, the prompts in version 2.2 do not use such revision prompts. Once the changes in the upstream codebase are stabilized, the Evalverse submodule will be subsequently updated.

Further, the H6 benchmark implemented in lm-eval-harness supports a wide range of evaluation options, some of which may affect the evaluation results as shown in Table 2. The table shows that the difference in the engine, dtype, and number of few-shot options can easily change the benchmark scores. Thus, in the H6 benchmark of Evalverse, we fix the number of few-shots for to those used in the Open LLM Leaderboard and use the “hf” engine and “float16” dtype exclusively.

Reproducibility. To ensure that the benchmark scores from the original repositories are reproducible with Evalverse, we evaluate various open source models using the original implementation and Evalverse and summarize the results in Table 3.

The table shows that benchmarks with little modification (H6, MT-Bench) produce same or almost same scores as the original implementation, as the evaluation is done by using the submodules that are the no or little modifications from the original implementation. We also note that the score differences in MT-Bench are from the randomness of using LLM-as-a-judge. On the other hand, the EQ-Bench benchmark results in a relatively larger score gap when compared to the original, due to the aforementioned intended modifications. We could not compare to the original IFEval, since its implementation contains only the core logic and data, without any evaluation scripts.

Evaluation speed. We also compare evaluation speed of using Evalverse with that of the original implementation in Table 4. The evaluation time with Evalverse and the original implementation for the H6, MT-Bench, EQ-Bench, and IFEval benchmarks using the SOLAR 10.7B Instruct model with 8×A100 GPUs. The H6 and MT-Bench have little evaluation time differences, whereas EQ-Bench evaluation time using Evalverse is faster for Evalverse. The main reason is the added data parallelism support in the Evalverse submodule.

Evaluation of Open Source Models In Table 5, multiple open source models are evaluated using Evalverse for H6, MT-Bench, EQ-Bench, and IFEval benchmarks, respectively. The evaluated models are divided into two categories of pre-trained and fine-tuned models. For pre-trained models, we measured H6 scores to assess the the base reasoning and knowledge capabilities of the models, while fine-tuned models were additionally evaluated on MT-Bench, EQ-Bench, and IFEval benchmarks to assess their multi-turn chat and instruction following ability. We used 8×A100 GPUs for evaluation, along with 8-bit quantization for larger models such as Mixtral 8×7B and Llama 2 70B.

Model	ARC	HellaSwag	MMLU	TruthfulQA	Winogrande	GSM8K	MT-Bench	EQ-Bench	IFEval
					Pre-trained Models				
Mistral 7B	61.43	83.31	62.64	42.62	79.16	37.83	-	-	-
Solar 10.7B	61.77	84.52	64.16	45.65	83.19	57.24	-	-	-
Yi 34B	65.44	85.75	76.51	56.27	83.19	65.73	-	-	-
Mixtral 8x7B	67.41	86.65	70.31	48.52	82.32	57.85	-	-	-
Llama 2 70B	67.58	87.00	68.83	44.81	83.35	52.62	-	-	-
Qwen 1.5 72B	66.21	85.97	77.25	59.57	82.72	68.69	-	-	-
					Fine-tuned Models				
Mistral 7B Instruct	63.65	84.63	59.10	66.81	78.93	41.85	7.600	66.57	0.5823
Solar 10.7B Instruct	71.42	88.20	65.28	71.71	83.19	67.40	7.580	73.34	0.5370
Yi 34B Chat	65.18	84.28	74.98	55.40	80.35	34.50	7.641	72.35	0.3577
Mixtral 8x7B Instruct	70.39	87.31	70.30	63.34	82.00	64.97	8.200	72.97	0.5850
Llama 2 70B Chat	65.36	85.72	62.70	53.09	79.72	52.84	7.142	70.14	0.5370
Qwen 1.5 72B Chat	67.58	86.28	77.70	63.11	79.72	29.11	8.347	82.81	0.6146

Table 5: Evaluation of open source models on various benchmarks using Evalverse.

5 Conclusion

We introduce Evalverse, a unified library for LLM evaluation that is easily expandable and accessible through no-code evaluation features. External benchmarks can be added via submodules, which makes addition of new benchmarks relatively easy while also making it possible for the added submodules to integrate upstream changes that may occur. Using communication platforms such as Slack, users can request evaluation jobs and query evaluation results via Slack messages, enabling a no-code LLM evaluations. We hope that by open-sourcing Evalverse, LLM evaluation can become more accessible and centralized, fueling further LLM development.

Acknowledgments

This work was supported by Institute of Information & Communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (No. RS-2024-00338140, Development of learning and utilization technology to reflect sustainability of generative language models and up-to-dateness over time).

Limitations

While Evalverse represents a significant step forward in the evaluation of Large Language Models (LLMs), there are inherent limitations to our approach. First, as the landscape of LLM evaluation is rapidly evolving, keeping Evalverse up-to-date with the latest tools and methodologies poses an ongoing commitment despite our best efforts to make the update process relatively easy. Second, while we aim to make the evaluation accessible via the no-code features in Evalverse, accurately interpreting the results may still require specialized knowledge. Additionally, our reliance on community contributions to expand and update the library could lead to disparities in the coverage of evaluation tools, potentially affecting the comprehensiveness of Evalverse. Lastly, while integrating with platforms like Slack enhances accessibility, it also introduces dependencies on third-party services, which may affect the long-term sustainability and adaptability of Evalverse.

Ethics Statement

In our Ethics Statement, we highlight the commitment of Evalverse to uphold ethical standards in the evaluation of Large Language Models (LLMs). We acknowledge the potential ethical issues, including privacy, security, and bias, associated with LLM evaluation. Evalverse is designed with a focus on transparency, accountability, and fairness, aiming to mitigate these concerns by promoting ethical research practices. This includes careful consideration of data sources, the impact on diverse communities, and efforts to reduce bias.

We stress the importance of responsible LLM use, advocating for evaluations that respect user privacy and data security. Evalverse is intended to foster an inclusive community of researchers by providing accessible evaluation tools and encouraging contributions from a broad spectrum of individuals. This approach not only addresses ethical concerns but also enhances the quality and inclusivity of LLM research. Our Ethics Statement reflects our dedication to advancing computational linguistics ethically, ensuring that LLM innovations consider their wider social and ethical impact.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Edward Beeching, Clémentine Fourrier, Nathan Habib, Sheon Han, Nathan Lambert, Nazneen Rajani, Omar Sanseviero, Lewis Tunstall, and Thomas Wolf. 2023. Open llm leaderboard. *Hugging Face*.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. 2023. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*.

Jiawei Chen, Hongyu Lin, Xianpei Han, and Le Sun. 2023. Benchmarking large language models in retrieval-augmented generation. *arXiv preprint arXiv:2309.01431*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

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. 2023. A framework for few-shot language model evaluation.

Neel Guha, Daniel E Ho, Julian Nyarko, and Christopher Ré. 2022. Legalbench: Prototyping a collaborative benchmark for legal reasoning. *arXiv preprint arXiv:2209.06120*.

Zishan Guo, Renren Jin, Chuang Liu, Yufei Huang, Dan Shi, Linhao Yu, Yan Liu, Jiaxuan Li, Bojian Xiong, Deyi Xiong, et al. 2023. Evaluating large language models: A comprehensive survey. *arXiv preprint arXiv:2310.19736*.

Muhammad Usman Hadi, Rizwan Qureshi, Abbas Shah, Muhammad Irfan, Anas Zafar, Muhammad Bilal Shaikh, Naveed Akhtar, Jia Wu, Seyedali Mirjalili, et al. 2023. A survey on large language models: Applications, challenges, limitations, and practical usage. *Authorea Preprints*.

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*.

Amr Hendy, Mohamed Abdelrehim, Amr Sharaf, Vikas Raunak, Mohamed Gabr, Hitokazu Matsushita, Young Jin Kim, Mohamed Afify, and Hany Hassan Awadalla. 2023. How good are gpt models at machine translation? a comprehensive evaluation. *arXiv preprint arXiv:2302.09210*.

Shashank Mohan Jain. 2022. Hugging face. In *Introduction to Transformers for NLP: With the Hugging Face Library and Models to Solve Problems*, pages 51–67. Springer.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Hanlei Jin, Yang Zhang, Dan Meng, Jun Wang, and Jinghua Tan. 2024. A comprehensive survey on process-oriented automatic text summarization with exploration of llm-based methods. *arXiv preprint arXiv:2403.02901*.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Dahyun Kim, Chanjun Park, Sanghoon Kim, Wonsung Lee, Wonho Song, Yunsu Kim, Hyeonwoo Kim, Yungi Kim, Hyeonju Lee, Jihoo Kim, Changbae Ahn, Seonghoon Yang, Sukyung Lee, Hyunbyung Park, Gyoungjin Gim, Mikyoung Cha, Hwalsuk Lee, and Sunghun Kim. 2023. Solar 10.7b: Scaling large language models with simple yet effective depth upscaling.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2021. Truthfulqa: Measuring how models mimic human falsehoods. *arXiv preprint arXiv:2109.07958*.

Darren Liu, Cheng Ding, Delgersuren Bold, Monique Bouvier, Jiaying Lu, Benjamin Shickel, Craig S Jabaley, Wenhui Zhang, Soojin Park, Michael J Young, et al. 2024. Evaluation of general large language models in contextually assessing semantic concepts extracted from adult critical care electronic health record notes. *arXiv preprint arXiv:2401.13588*.

OpenAI. 2023. Gpt-4 technical report.

Samuel J Paech. 2023. Eq-bench: An emotional intelligence benchmark for large language models. *arXiv preprint arXiv:2312.06281*.

Philip Resnik and Jimmy Lin. 2010. Evaluation of nlp systems. *The handbook of computational linguistics and natural language processing*, pages 271–295.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.

Karan Singhal, Shekoofeh Azizi, Tao Tu, S Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, et al. 2023. Large language models encode clinical knowledge. *Nature*, 620(7972):172–180.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Neng Wang, Hongyang Yang, and Christina Dan Wang. 2023. Fingpt: Instruction tuning benchmark for open-source large language models in financial datasets. *arXiv preprint arXiv:2310.04793*.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.

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*.

Congying Xia, Chen Xing, Jiangshu Du, Xinyi Yang, Yihao Feng, Ran Xu, Wenpeng Yin, and Caiming Xiong. 2024. Fofo: A benchmark to evaluate llms' format-following capability. *arXiv preprint arXiv:2402.18667*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2024. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36.

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*.

Yuchen Zhuang, Yue Yu, Kuan Wang, Haotian Sun, and Chao Zhang. 2024. Toolqa: A dataset for llm question answering with external tools. *Advances in Neural Information Processing Systems*, 36.