POKT AI Lab

Report IV

Authors:
Nicolas Aguirre (nicolas@poktscan.com)
Ramiro Rodríguez Colmeiro (ramiro@poktscan.com)
POKTscan Data Science Team

July 1, 2024

Disclaimer

This work was partially funded by Pocket Scan Technologies LLC, a company operating on the Pocket Network.

Changelog

v1.0	2024-07-01	First version.

Table of Contents

1 Progress Overview

During the month of June, the socket achieved the following milestones:

- Merged issues on the Test-Bench code:
  - **Manager**:
    - Add same trigger restrictions based on blocks. [1]
    - Create logic for tokenizer signature task. [2]
  - **Sampler**:
    - Create logic for tokenizer signature task. [3]
    - Fix generation_until in LM classes. [4]
  - **Evaluator**:
    - Create initial code for lmeh processing. [5]
    - Add signatures tokenizer evaluation. [6]
    - Finish code for lmeh. [7]
  - **Sidecard**:
    - Added endpoint for tokenizer in llm nodes. [8]
  - **General**:
    - Modularizing the tasks to allow signatures. [9]
    - Task configs: update metrics and filters. [10]
    - Added website to show metrics. [11]
    - Full ML Test bench integration. [12]

In June 2024, significant progress was made on the socket, resulting in the completion of various tasks and milestones. Finally the different components of the Machine Learning Test-Bench (MLTB), such as the Manager, Sampler, Evaluator, and Sidecard were integrated. This integration culminated presenting the leaderboards on a website considering the results of the corresponding task metrics.

In the next sections of the current report the details of these developments and their impli-

[1] https://github.com/pokt-scan/pocket-ml-testbench/pull/67
[2] https://github.com/pokt-scan/pocket-ml-testbench/pull/62
[3] https://github.com/pokt-scan/pocket-ml-testbench/pull/57
[4] https://github.com/pokt-scan/pocket-ml-testbench/pull/78
[5] https://github.com/pokt-scan/pocket-ml-testbench/pull/60
[6] https://github.com/pokt-scan/pocket-ml-testbench/pull/63
[7] https://github.com/pokt-scan/pocket-ml-testbench/pull/65
[8] https://github.com/pokt-scan/pocket-ml-testbench/pull/56
[9] https://github.com/pokt-scan/pocket-ml-testbench/pull/54
[10] https://github.com/pokt-scan/pocket-ml-testbench/pull/80
[11] https://github.com/pokt-scan/pocket-ml-testbench/pull/81
[12] https://github.com/pokt-scan/pocket-ml-testbench/pull/83

cations are presented. These sections provide an overview of the completed work, updating components initially considered, and the progress made towards the final goal of the socket.

Furthermore, looking ahead, there are several future work ideas to consider. These include refining the evaluation metrics further, exploring additional benchmark for extended functionalities. These future initiatives aim to enhance the network capacity guided by the state-of-the-art in the field of machine learning and blockchain technology.

Note: This report was written July 1, 2024, and the Huggingface announced an update in the Huggingface Open Language Model Leaderboard (HFOLML) at Jun 26, 2024. While having an exact copy of HFOLML was not the final objective of the MLTB, it is a way to give the machine learning (ML) community an easy comparison of the POKT Network models quality against known models. The major differences of this new HFOLML compared to the one implemented in the MLTB are:

- Some new tasks need to be implemented.
- A new metric needs to be coded.
- Changes in supported tasks configs.

This report will not cover the new Leaderboard and we will all these changes as future work.

Nevertheles, at POKTscan and PNYX, we are confident that the MLTB is ready to be developed in order to reproduce the **old** Leaderboard. furthermore, we except that the MLTB will be able in a short time to be updated to the new Leaderboard.

Addressing evaluations of language model (LM), a topic that is at the forefront of knowledge, has an inherently dynamic nature. That is to say, the constant (and increasingly accelerated) improvement of the LM requires the development of new evaluations that have to be constantly improved. What happened with the HFOLML is just the first case the POKT Network faces when it wants to support AI in it. At POKTscan we alert the community about this inevitable game of cat and mouse, and we encourage everyone to join the game.

2 Machine Learning Test-Bench

"Thinking, analyzing, inventing are not anomalous acts, they are the normal breathing of intelligence. To glorify the occasional fulfillment of that function, to treasure old and foreign thoughts, to recall with incredulous amazement what the doctor universalis thought, is to confess our languor or our barbarism. Every man must be capable of all ideas, and I understand that in the future he will be."

"Pierre Menard, author of Don Quixote", "Ficciones", Jorge Luis Borges

Throughout these four months, POKTscan and PNYX have worked on reproducing the HFOLML [12] that serves as a reference for the POKT Network community and relates its model's offer to terms that are familiar for the ML community. With this document we conclude our series of reports. The reports need to be considered as a whole to understand the full picture of the work done.

Throughout this section we will summarize the different modules:

- **Manager**
- **Regisrter**
- **Sampler**
- **Requester**
- **Evaluator**

We will show how they interact, as detailed from Figure 1 to 9. It is important to mention that we will not delve into some of the design decisions that were made, since most of these were discussed in the previous reports [13], which we refer the reader to for more details. Finally, the results obtained when evaluating some LMs will be presented.

2.1 Workflows

Consider a scenario where N nodes, serving unknown LMs, are staked in service S. The MLTB is configured to track M metrics (i.e. all the HFOLML metrics) over all nodes N staked in S. The MLTB has access to a given number of POKT apps A, also staked in S.

Then, the main steps are the following:

[13] https://github.com/pokt-scan/pocket-ml-testbench/tree/main/reports

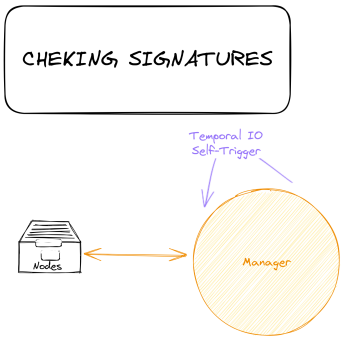

Figure 1

Checking signatures: The Manager will check if N in S has a signature σ associated in the nodes collection. The signatures are the first thing to check because they are either requirements for other tasks (i.e. tokenizers data) or they signal if a node changed or not (which would trigger complete resampling).

In the particular case of testing LM, the most important signature consists of a hash of its tokenizer T, hereafter referenced as $\sigma(T)$. In ease of expression, we will refer to the signature of node N in S as $\sigma(T_N^S)$. Without a tokenizer signature (signaling that we know the tokenizer) the rest of the tests cannot be performed. Figure 1 shows the workflow until this step.

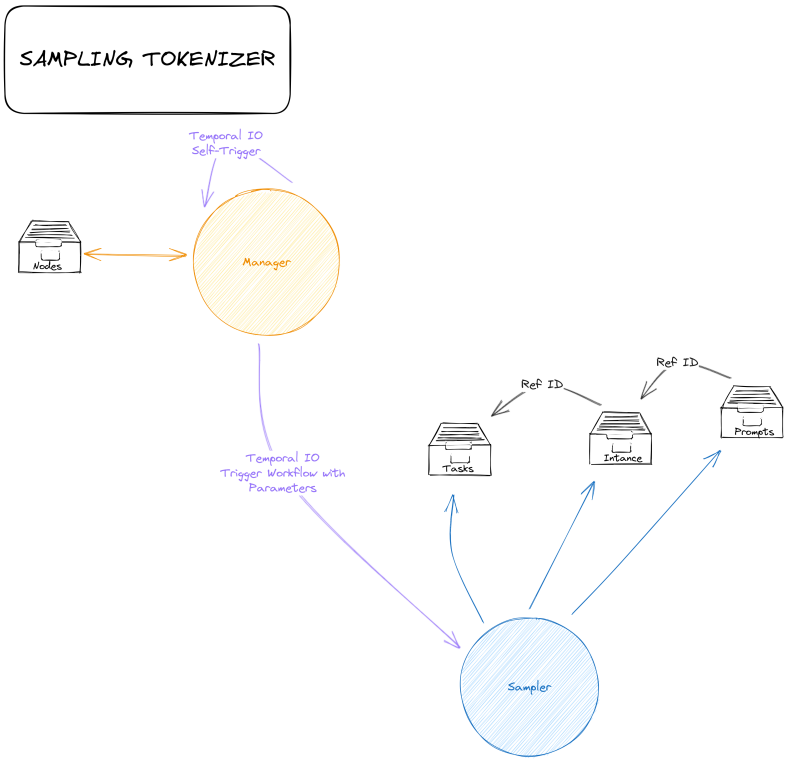

Figure 2

Sample tokenizer: To produce the tokenizer signature, the Manager triggers the Sampler module to save a prompt for the Requester pointing to the endpoint of N_s in the *prompts* collection. Figure 2 shows the workflow until this step.

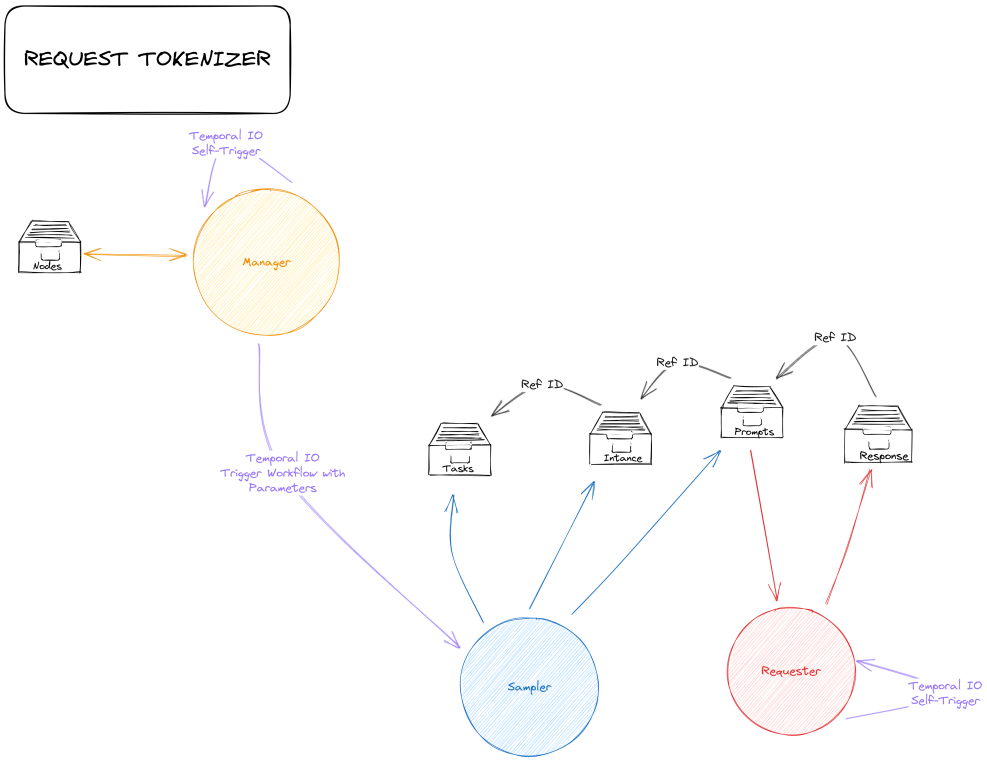

Figure 3

Request tokenizer: The tokenizer signature, is produced from the a full json tokenizer T data that is available in the *pokt/tokenizer* endpoint of the POKT Sidecar. This prompt is not responded by the OpenAI API because it is not part of the spec since the tokenizer is known. In the case of the POKT network, the tokenizer is unknown because the model itself is unknown. Figure 3 shows the workflow until this step.

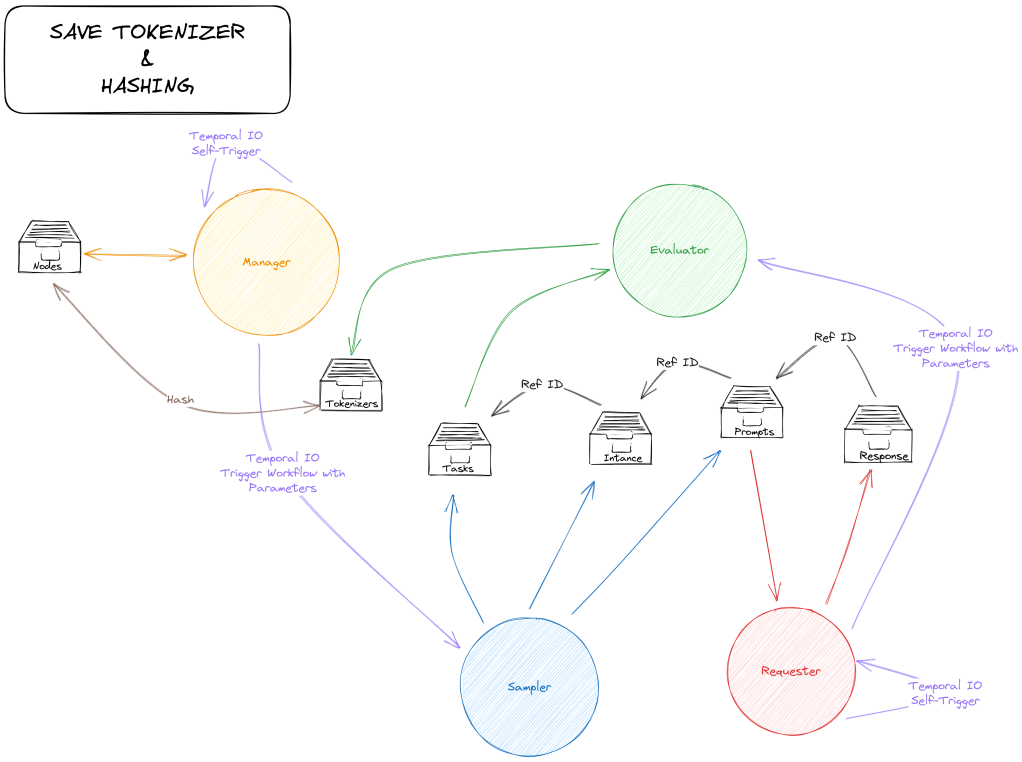

Figure 4

Save tokenizer: Finally the evaluator loads the Requester response and calculates the tokenizer hash. If the tokenizer is not already tracked in the tokenizers collection, it will create a new entry. Finally the Manager will recover the tokenizer hash from the results collection entry and add it to the node data. Figure 4 shows the workflow until this step.

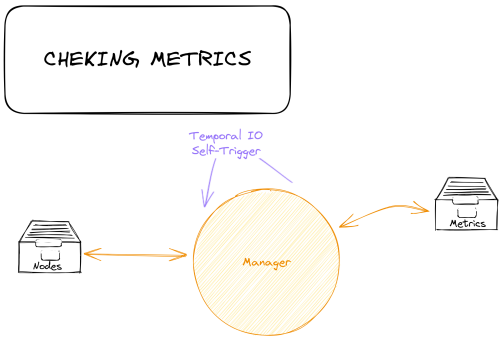

Figure 5

Checking metrics: Once a node has its signature, the Manager will check if any sample related to a metric of a task needs to be updated, either because the configured time has expired, or it does not yet have a sufficient number of samples to produce reliable metrics. If required, the Manager will trigger the Sampler, requiring from the Sampler a quantity q of samples for the task. Figure 5 shows the workflow until this step.

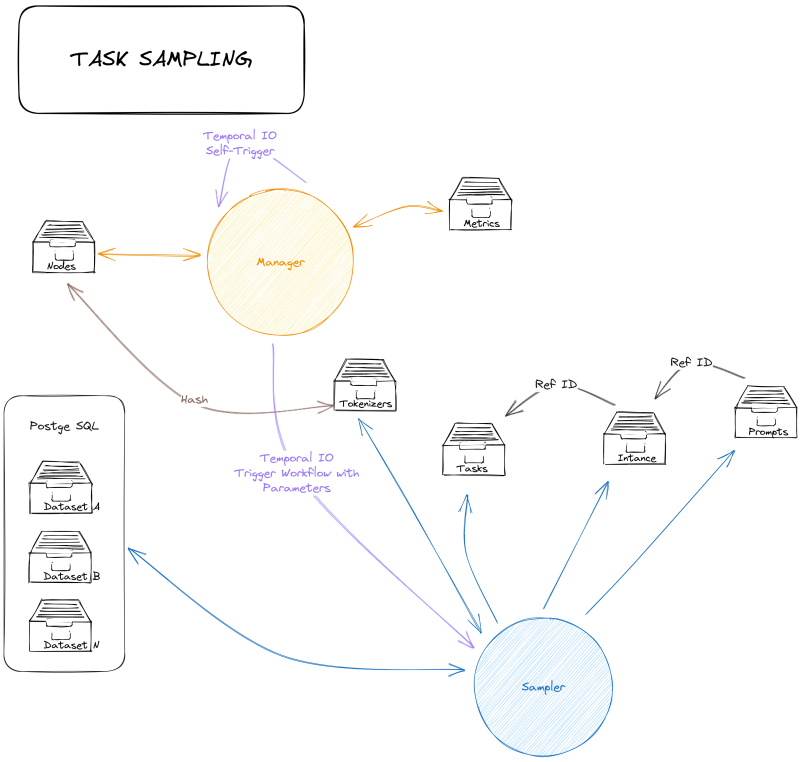

Figure 6

Task sapling: When the Sampler receives the request from the Manager, it will load the dataset corresponding to the task but avoiding to load samples previously requested from the evaluation dataset. The Sampler randomly selects q docs from the dataset, and create a LM prompt following the OpenAI API. The Sampler ends by saving the task, instance, prompt and request information the corresponding collections. Figure 6 shows the workflow until this step.

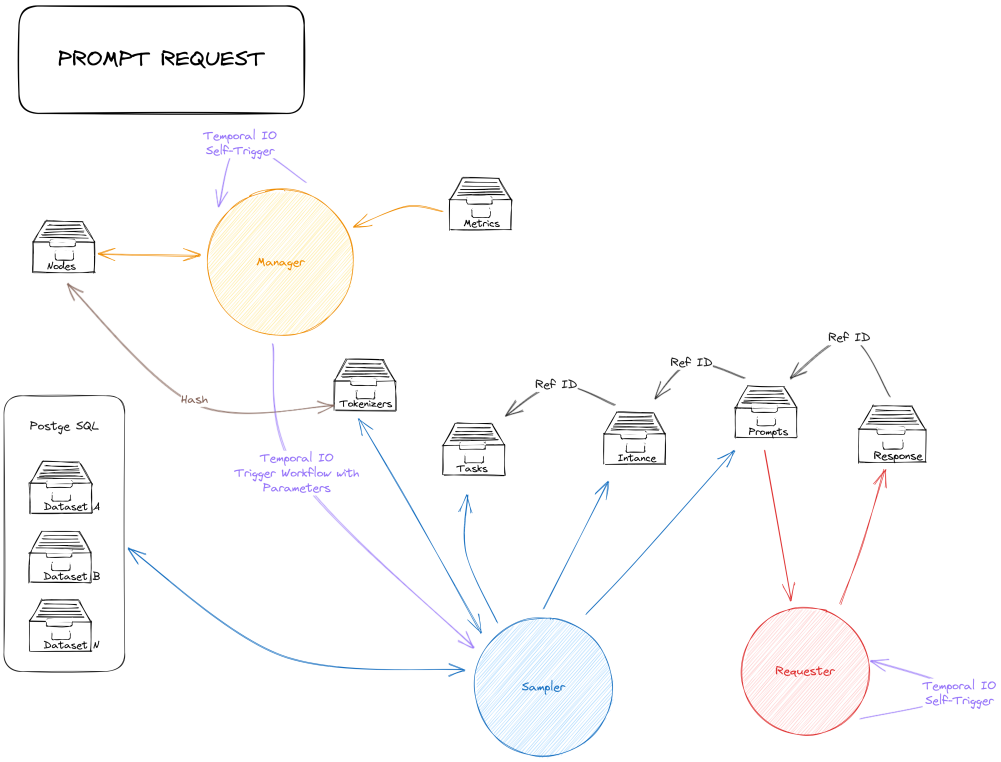

Figure 7

Prompt request: After the asynchronous creation of the request by the Sampler, the Requester will try to prompt the node whenever it enters into session with one of its controlled apps. It is normal that the task will remain in the collection for a long period until this happens. Once the requested node is found in session, the relay is done and the result is written to the response collection. Figure 7 shows the workflow until this step.

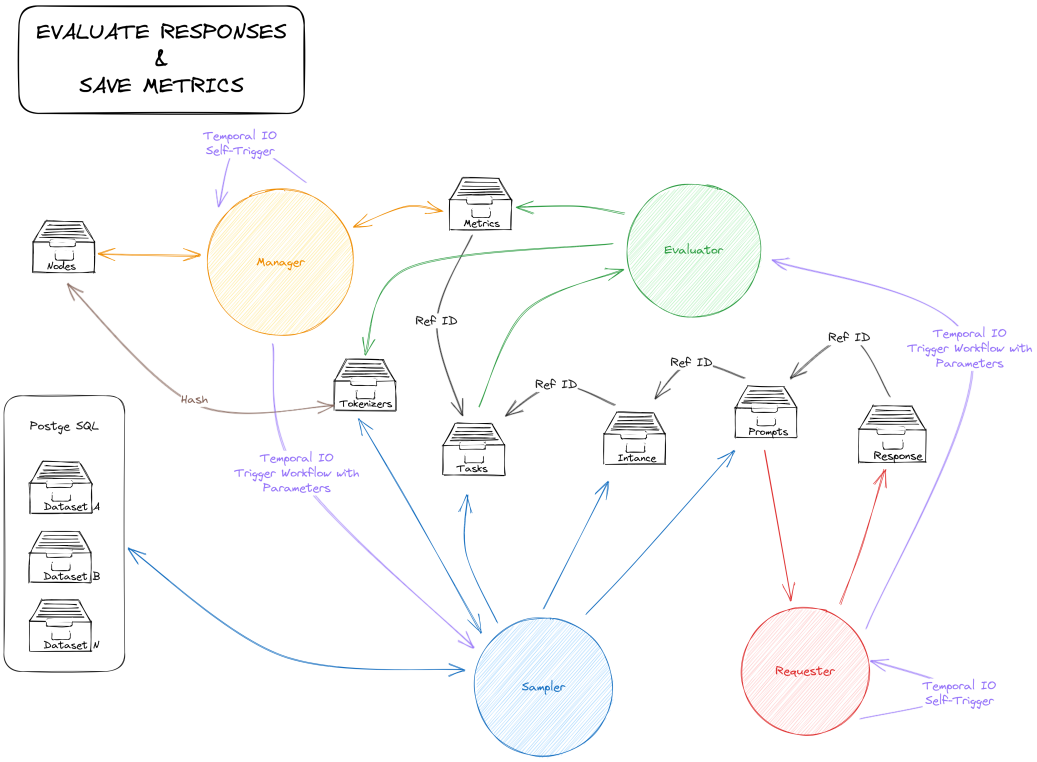

Figure 8

Evaluation response & save metrics: The Evaluator can be triggered by the Requester when a task is marked as done, meaning that all request had been responded, or by a periodic time previously configured. At this point it is necessary to regenerate the same evaluation state as if lm-eval-harness had been executed locally, that is to say, the same execution state is generated (this includes: task, instance, prompt, datasets, etc) as the Sampler, with the addition that now we have the responses. It is important to mention that because Requester simply returns any response that the Pocket Network node sends it, and this includes any type of error, the Evaluator will only be able to continue with the evaluation and generate metrics when at least one sample has all its requests responded. To recall, the relation between a sample and requests is 1:N, as was illustrated in the previous report. Figure 8 shows the workflow until this step.

Leaderboard Update: Finally, after the Manager has written all the metrics data in the corresponding nodes collections, the API process will re-process the higher level information, such as the leaderboard scores. Figure 9 shows the last step of the workflow.

2.2 Experiments

In order to asses if the proposed replication of the HFOLML was correct, we selected three models with different characteristics and compared them to the publicized scores. The selected models were:

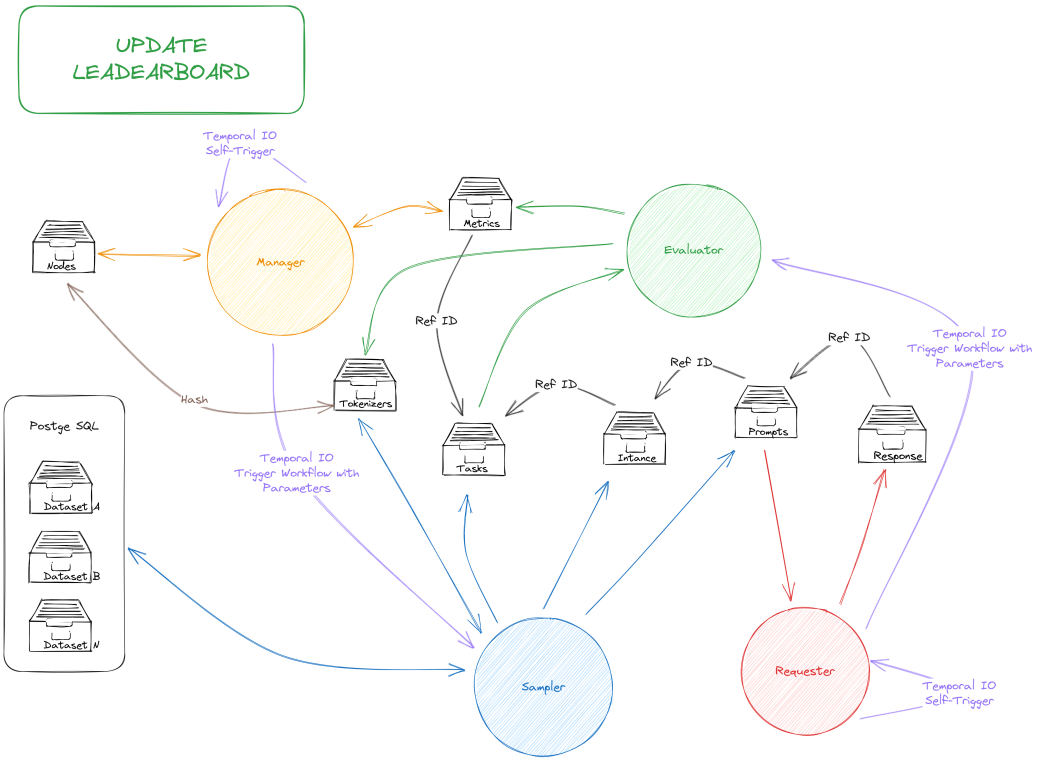

Figure 9: Update metrics.

- Llama-3-8b-instruct [14]
- Deepseek-coder-6.7B-instruct [15]
- TinyLlama-1.1B-Chat-v1.0 [16]

Models were evaluated in the same tasks as the ones proposed in the HFOLML. These are the following:

- ARC [4]
- Hellaswag [19]
- MMLU [7]
- TruthfulQA [10]
- Winogrande [15]
- GSM8K [5]

Since at the moment of the experiment we are dealing with live endpoints that have an associated cost, we do not sample the whole datasets, instead we sample only 50 samples for each task (or sub-task in the case of MMLU). The effect on the tests accuracy according to Polo et al. should be less than 5% [14]. Regarding the Language Model Evaluation Har-

[14] https://huggingface.co/casperhansen/llama-3-8b-instruct-awq
[15] https://huggingface.co/TheBloke/deepseek-coder-6.7B-instruct-AWQ
[16] https://huggingface.co/TheBloke/TinyLlama-1.1B-Chat-v1.0-AWQ

ness (LMEH) [2] code, the MLTB implements the release `0.4.2`, this is not the same as the HFOLML due to a major refactoring was performed after Huggingface released his Leaderboard. This decision was made to improve maintainability of the repository provided and trying to assessing correctly any discrepancies in metrics or configs between de version using by Huggingface and the LMEH.

Regarding the command to run the evaluation, it was done using the *morse-poc* deployment files. It requires only a few steps [17]: writing a `.env` with necessary variables placed with the docker compose defined next and executing the following command:

```
cd docker-compose/morse-poc
docker-compose up -d
```

Listing 1: Command to run the evaluation.

Internally, the command will run the following containers:

- a vLLM [9] engine for serving a model,
- components related to Temporal server,
- Manager, Register, Sampler, Requester, and Evaluator modules as Temporal workers,
- the pocketnetwork protocaol, its genesis, and three pocket network lean nodes,
- components related to PostgreSQL database to store the datasets corresponding to each task,
- components related to MongoDB to store the rest of the data,
- a minimal web site an API to display results as they are collected.

2.3 Results

In Table 1 we present the results obtained when evaluating the presented LMs. The table shows the Pearson correlation coefficient R and its p-value for each pair of node/model and task, and the average of the metrics obtained both for the current node and the HFOLML. Of course, in a production scenario, we would have more nodes and no information about the ground truth, since it is not possible to kwon the real model behind the node. Nevertheless, it can be seen that the R is high (near 1.0 which is the ideal) and the p-value shows that the results have statistical significance in two of the three nodes [18],

We also present the results for each model in particular in Figures 10, 11, and 12.

[17] We refer the reader to https://github.com/pokt-scan/pocket-ml-testbench/tree/main/docker-compose/morse-poc for more instructions on how to deploy.
[18] The node that has a p-value above 0.05 is actually very near (0.06) and is also the worst performing model.

Table 1: Results obtained with the MLTB and those published in the HFOLML.

Node	Model	R	p	Average		ARC		Hellaswag		MMLU		TruthfulQA		Winogrande		GSM8K	
				MLTB	HFOLML	MLTB	HFOLML	MLTB	HFOLML	MLTB	HFOLML	MLTB	HFOLML	MLTB	HFOLML	MLTB	HFOLML
A	Llama-3-8b-instruct	0.83	< 0.05	69.8	66.9	54.0	60.7	80.0	78.5	63.3	67.1	58.6	51.6	88.0	74.5	74.0	68.7
B	Deepseek-coder-6.7B-instruct	0.92	< 0.05	48.3	43.6	41.9	38.1	56.0	55.1	39.9	39.0	43.5	45.6	60.0	56.8	0.0	26.8
C	TinyLlama-1.1B-Chat-v1.0	0.78	0.06	30.4	37.3	25.9	36.1	31.9	61.1	24.0	25.4	50.4	37.5	50.0	61.2	0.0	2.3

R: Pearson correlation coefficient. p: p-value.

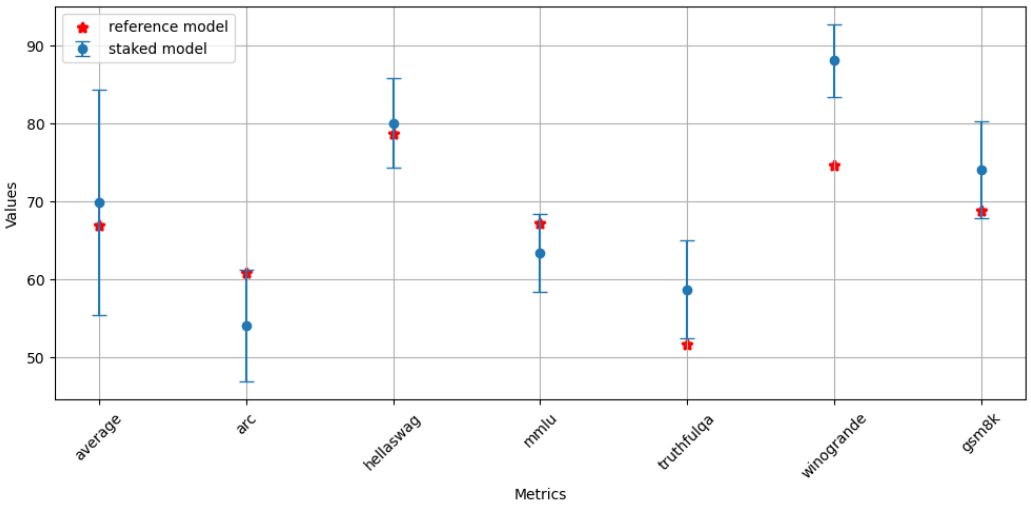

Figure 10: Model: Llama-3-8b-instruct; Correlation Coef: 0.833 ; P-value: 0.039

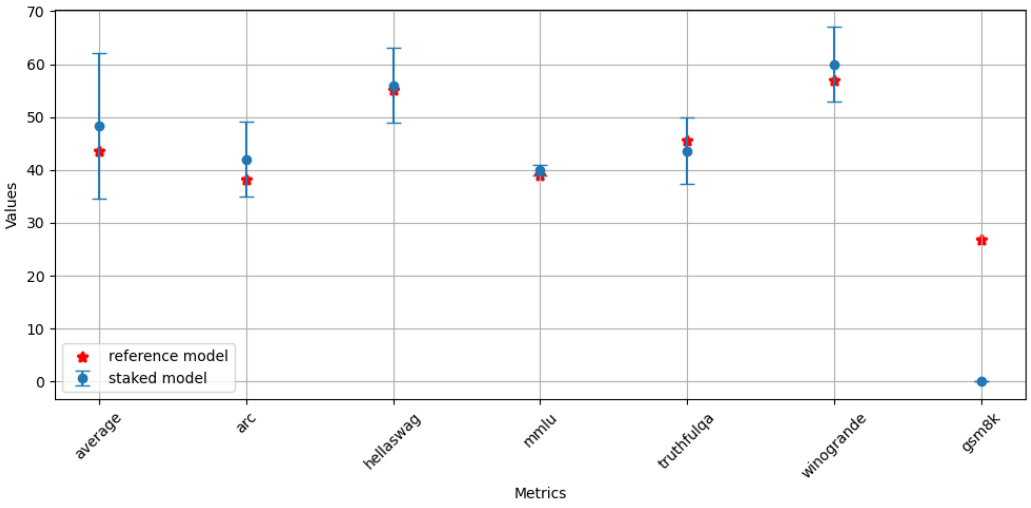

Figure 11: Model: Deepseek-coder-6.7B-instruct; Correlation Coef: 0.924 ; P-value: 0.009

2.4 Conclusions

The MLTB was developed to perform arbitrary tests on ML models that are staked in the POKT Network. In this first iteration it was configured to recreate the HFOLML, a known source for model quality in the ML community. After the initial tests we can say that the MLTB is able to deliver on its promises: it was successfully deployed on a POKT Network local net, communicated with POKT nodes hosting LMs and successfully retrieved the required data to

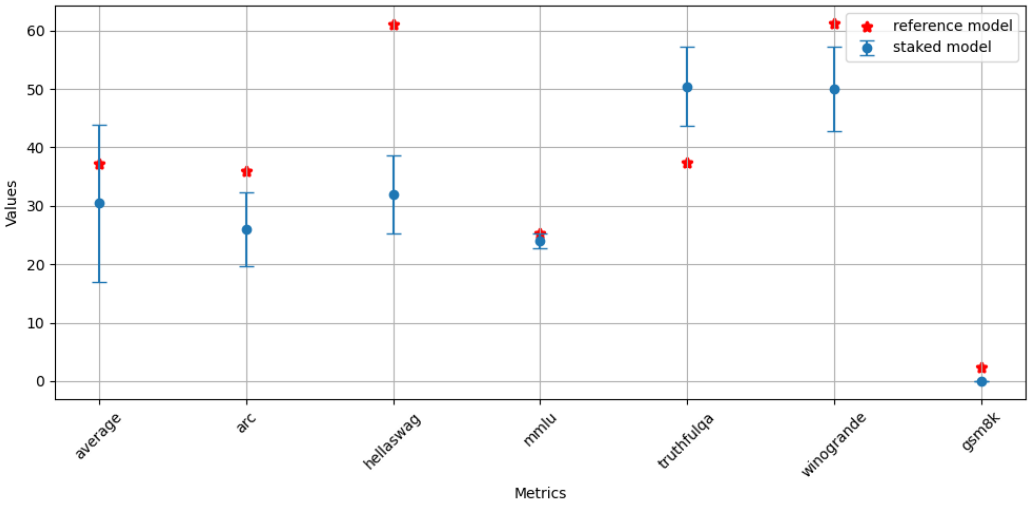

Figure 12: Model: TinyLlama-1.1B-Chat-v1.0; Correlation Coef: 0.785 ; P-value: 0.064

populate the leaderboard.

The MLTB is not only able to reproduce the HFOLML data, it also does it with much less samples (and hence cost) and keep high correlation with fully-sampled tests. Also, the test bench is created to be fully modular and scalable, being able to process tests form many nodes concurretly, something beyond the capabilities of the LMEH that was not created to support loads as the ones we expect in the POKT Network.

Regarding the update in the HFOLML, it is important to recall what Borges said: "Thinking, analyzing, inventing are not anomalous acts, they are the normal breathing of intelligence". As the ML field progresses, the adoption and refinement of frameworks like the LMEH will remain crucial, ensuring that benchmarks evolve in tandem with the technologies they aim to measure. In the context of the POKT network, MLTB is a first step towards the adoption of a decentralized ML market following the state of the art in the field of benchmarking, and thus trying to shed light on the quality of service expected.

3 Future Work

So we have created a scalable and organized LM evaluation workflow. But does this have to end here? What other possibilities are open?

We believe that the methodology presented throughout the different reports is abstract enough to be able to implement many ideas on top of the MLTB. Our main ambitions are the following:

More leaderboards

The HFOLML is not the only leaderboard out there, there are many other leaderboards that could be interesting to replicate. For example, more specific evaluations could be added for LMs, such as:

1. The so-called "function call evaluation" [13].
2. The Massive Text Embedding Benchmark (MTEB) Leaderboard [11].
3. The LLM Safety Leaderboard [16].

On the other hand, if the network wants to add support for multimodal models, specific benchmarks [18] should be adapted for these cases.

More Tests

Also we can develop custom test, or generative tests to counter the overspecialization (or contamination) of new models in the mainstream tests. The last can alert about data leakage, a common problem during the traing of LMs [20, 6, 17, 1], both closed and open source.

Model Signatures

Getting signatures from a model is not restricted to getting a tokenizer. The concept of a signature is to identify if a model is of a given kind and track its changes. We might not be able to tell (confidently) the name of the model being staked in the POKT network but we can find all models of the same kind and group them by means of random signatures.

Also there are techniques likes watermarking [8] that could be useful to implement in POKT and the MLTB can be used to calculate and track such signatures.

On-chain

The ideal ML test suite should be one developed by a big community and implemented by parties with no conflicts of interest [19]. We consider this work as an actionable first step towards the descentralization of ML tests, and we want to explore ways of enabling a crypto-economic incentive for providing this service, something close to the phylosofy of the POKT Network watchers [20].

[19] Model providers cannot be trusted to provide service with models that are the same as what they publish in journals [3].

[20] For reference, a description of a watcher actor can be found here: https://github.com/pokt-network/pocket-network-protocol/blob/main/utility/README.md#33-fisherman-protocol

List of Acronyms

HFOLML	Huggingface Open Language Model Leaderboard
LM	language model
LMEH	Language Model Evaluation Harness
ML	machine learning
MLTB	Machine Learning Test-Bench

Reference List

[1] Simone Balloccu et al. *Leak, Cheat, Repeat: Data Contamination and Evaluation Malpractices in Closed-Source LLMs*. Feb. 22, 2024. DOI: 10.48550/arXiv.2402.03927. arXiv: 2402.03927[cs]. URL: http://arxiv.org/abs/2402.03927 (visited on 07/01/2024).

[2] Stella Biderman et al. *Lessons from the Trenches on Reproducible Evaluation of Language Models*. May 23, 2024. DOI: 10.48550/arXiv.2405.14782. arXiv: 2405.14782[cs]. URL: http://arxiv.org/abs/2405.14782 (visited on 05/29/2024).

[3] Lingjiao Chen, Matei Zaharia, and James Zou. *How is ChatGPT's behavior changing over time?* Oct. 31, 2023. arXiv: 2307.09009[cs]. URL: http://arxiv.org/abs/2307.09009 (visited on 07/01/2024).

[4] Peter Clark et al. "Think you have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge". In: *ArXiv* abs/1803.05457 (2018).

[5] Karl Cobbe et al. *Training Verifiers to Solve Math Word Problems*. 2021. arXiv: 2110.14168 [cs.LG].

[6] Shahriar Golchin and Mihai Surdeanu. *Data Contamination Quiz: A Tool to Detect and Estimate Contamination in Large Language Models*. May 24, 2024. DOI: 10.48550/arXiv.2311.06233. arXiv: 2311.06233[cs]. URL: http://arxiv.org/abs/2311.06233 (visited on 07/01/2024).

[7] Dan Hendrycks et al. *Measuring Massive Multitask Language Understanding*. Jan. 12, 2021. arXiv: 2009.03300[cs]. URL: http://arxiv.org/abs/2009.03300 (visited on 07/01/2024).

[8] John Kirchenbauer et al. "A watermark for large language models". In: *International Conference on Machine Learning*. PMLR. 2023, pp. 17061–17084.

[9] Woosuk Kwon et al. *Efficient Memory Management for Large Language Model Serving with PagedAttention*. Sept. 12, 2023. DOI: 10.48550/arXiv.2309.06180. arXiv: 2309.06180[cs]. URL: http://arxiv.org/abs/2309.06180 (visited on 06/27/2024).

[10] Stephanie Lin, Jacob Hilton, and Owain Evans. "TruthfulQA: Measuring How Models Mimic Human Falsehoods". In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 3214–3252. DOI: 10.18653/v1/2022.acl-long.229. URL: https://aclanthology.org/2022.acl-long.229.

[11] Niklas Muennighoff et al. *MTEB: Massive Text Embedding Benchmark*. Mar. 19, 2023. DOI: 10.48550/arXiv.2210.07316. arXiv: 2210.07316[cs]. URL: http://arxiv.org/abs/2210.07316 (visited on 07/01/2024).

[12] *Open LLM Leaderboard - a Hugging Face Space by open-llm-leaderboard*. URL: https://huggingface.co/spaces/open-llm-leaderboard/open_llm_leaderboard (visited on 06/25/2024).

[13] Shishir G. Patil et al. *Gorilla: Large Language Model Connected with Massive APIs*. May 24, 2023. arXiv: 2305.15334[cs]. URL: http://arxiv.org/abs/2305.15334 (visited on 06/25/2024).

[14] Felipe Maia Polo et al. *tinyBenchmarks: evaluating LLMs with fewer examples*. May 26, 2024. arXiv: 2402.14992[cs, stat]. URL: http://arxiv.org/abs/2402.14992 (visited on 06/21/2024).

[15] Keisuke Sakaguchi et al. "WinoGrande: An Adversarial Winograd Schema Challenge at Scale". In: *arXiv preprint arXiv:1907.10641* (2019).

[16] Boxin Wang et al. *DecodingTrust: A Comprehensive Assessment of Trustworthiness in GPT Models*. Feb. 26, 2024. DOI: 10.48550/arXiv.2306.11698. arXiv: 2306.11698[cs]. URL: http://arxiv.org/abs/2306.11698 (visited on 07/01/2024).

[17] Ruijie Xu et al. *Benchmarking Benchmark Leakage in Large Language Models*. Apr. 29, 2024. DOI: 10.48550/arXiv.2404.18824. arXiv: 2404.18824[cs]. URL: http://arxiv.org/abs/2404.18824 (visited on 07/01/2024).

[18] Xiang Yue et al. *MMMU: A Massive Multi-discipline Multimodal Understanding and Reasoning Benchmark for Expert AGI*. June 13, 2024. arXiv: 2311.16502[cs]. URL: http://arxiv.org/abs/2311.16502 (visited on 06/25/2024).

[19] Rowan Zellers et al. "HellaSwag: Can a Machine Really Finish Your Sentence?" In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019.

[20] Hugh Zhang et al. *A Careful Examination of Large Language Model Performance on Grade School Arithmetic*. May 3, 2024. DOI: 10.48550/arXiv.2405.00332. arXiv: 2405.00332[cs]. URL: http://arxiv.org/abs/2405.00332 (visited on 07/01/2024).