Pharo-LLM: A call for a sovereign LLM ecosystem in Pharo

From dependency to autonomy: a full control

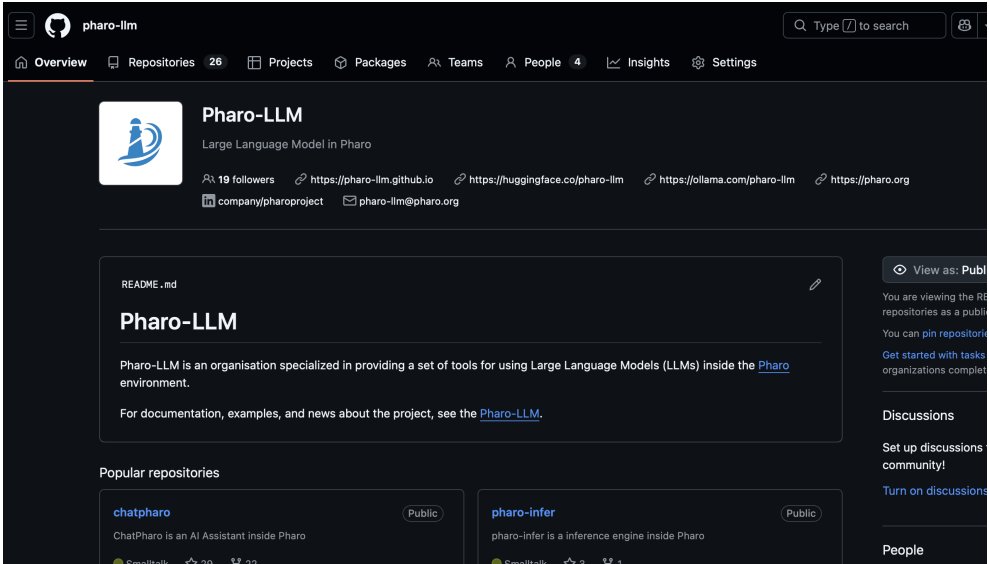
Omar Abedelkader

2th July 2026

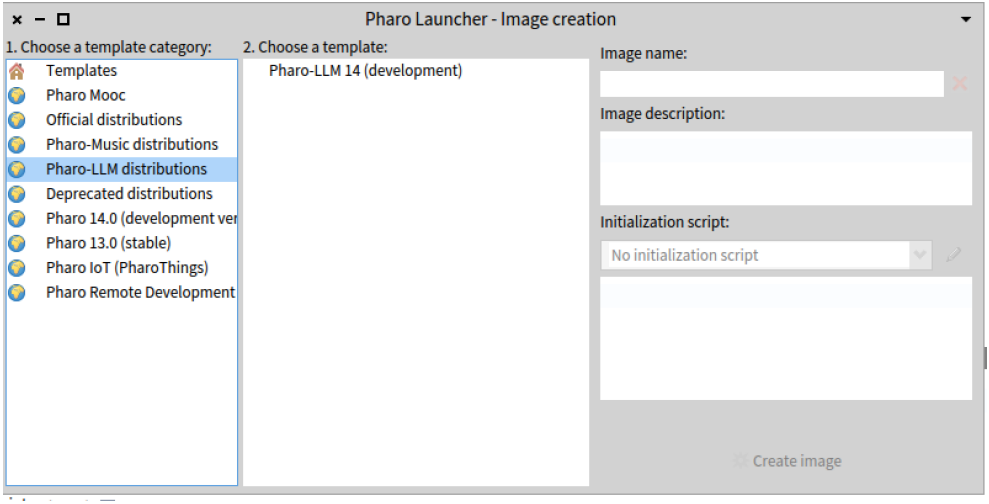
Who Am I ?

- PhD Student in Computer Science at University of Lille, France.
- Code Completion, Code Generation

What is Pharo-LLM?



What is Pharo-LLM?



Overview

- What is Pharo-LLM?
- Concrete Example of Pharo-LLM
- Pharo-LLM vision

What is the Pharo-LLM?

- Set of tools and libraries to integrate LLMs in Pharo.
- Creating, running, and interacting directly inside Pharo.
- Native tools instead of disconnected wrappers.
- Integrated with objects, inspectors, debuggers, and workflows.
- Built to be extended by the Pharo community

The current Tools in Pharo-LLM

- ChatPharo: Conversational interface inside Pharo.
- Pharo-copilot: Intelligent Completion Engine
- Pharo-Infer: Inference Engine for LLMs in Pharo.
- Pharo-MCP: MCP Protocol implemented in Pharo.
- Pharo-HuggingFace: an Interface to talk with Hugging Face
- And More ....

Real AI Cost

- Meta is paying a bill of 221 Million dollars for using Claude (73 T tokens) per month.
- In one single year Meta is paying a bill of 2.65 Billion $ for using Claude.
- Microsoft is paying a bill of 500 million dollars for using Claude.

How can we have a sovereign LLM ecosystem in Pharo?

- Having the tools inside Pharo to create, run, and interact with LLMs.
- Having the ability to train and fine-tune our own models.
- Having the ability to deploy and maintain our own models.

How can we have a sovereign LLM ecosystem in Pharo?

- Having the tools inside Pharo to create, run, and interact with LLMs.
- Having the ability to train and fine-tune our own models.
- Having the ability to deploy and maintain our own models.

How can we have a sovereign LLM ecosystem in Pharo?

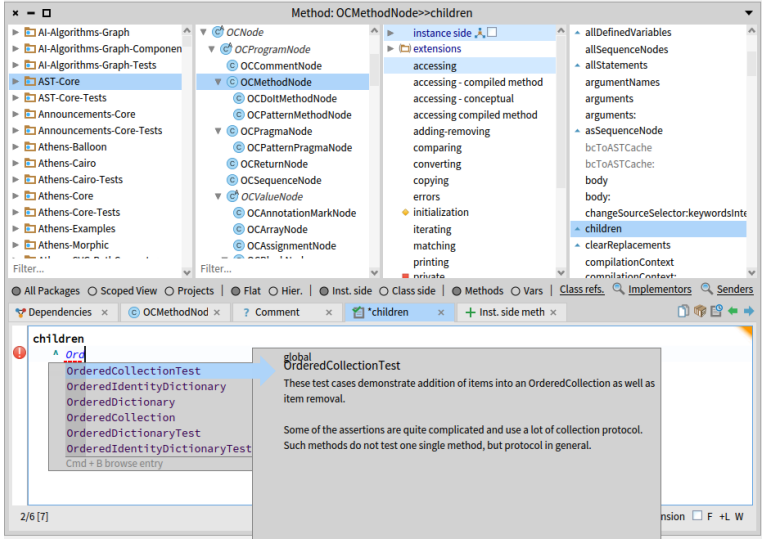
- Having the tools inside Pharo to create, run, and interact with LLMs.
- Having the ability to train and fine-tune our own models.
- Having the ability to deploy and maintain our own models.

How can we train/fine tune our models?

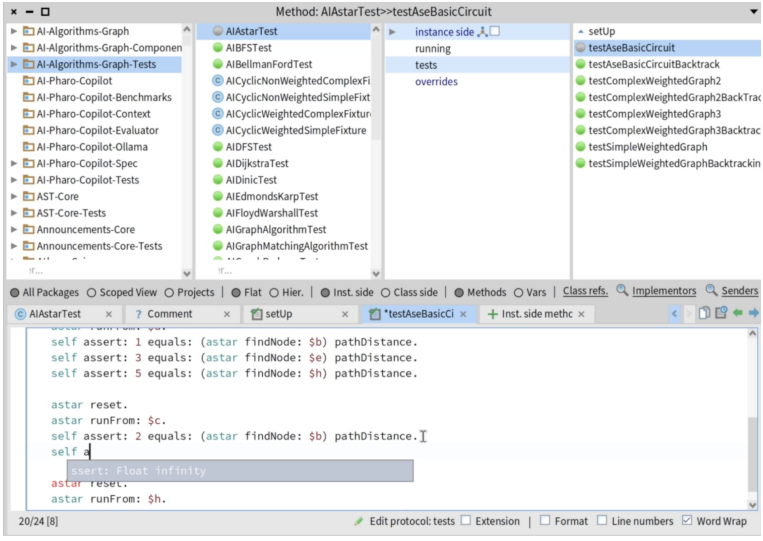
A Concrete Example: Multi-token Code Completion

- Pharo-Copilot is a code completion engine that uses LLMs to predict the next Pharo code tokens.
- Predict the next Pharo code tokens, not only one word.

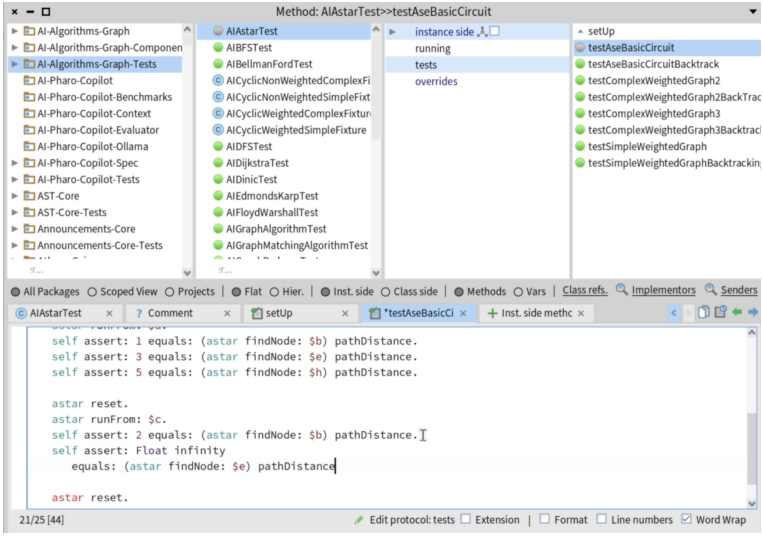
Single Token Completion



Multi-token Completion



Multi-token Completion



Form to participate in the Pharo-Copilot Betas

How to train/fine-tune our models?

- Open-weight code LLMs (Qwen, DeepSeek, etc..)
- The approach uses two steps:
  - continued pre-training to teach Pharo syntax and APIs;
  - supervised fine-tuning to adapt the model to code completion.
- Qwen2.5-Coder and Mellum models.
- Small models from 0.5B to 7B parameters were enough to obtain strong results.

How did we get data?

- We collected Pharo repositories from GitHub ($\sim$ 1200).
- We kept repositories with:
  - MIT license ($\sim$ 748);
  - Tonel format;
  - compatibility with Pharo 10 or newer ($\sim$ 415).

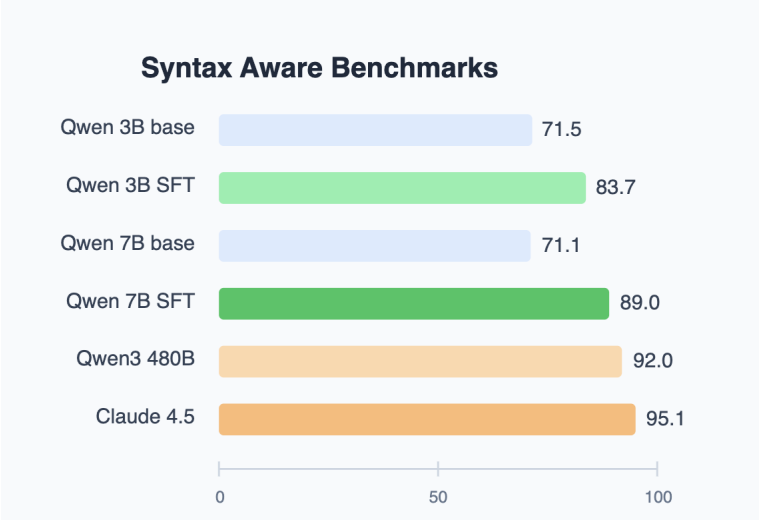
How much did it take?

Stage	Model Size	Duration
Continuous Pre-training (Training)	0.5B–1.5B	2–3 days
	3B–7B	5–6 days
Supervised Fine-tuning (Fine-tuning)	0.5B–1.5B	1–1.5 days
	3B–7B	2–3 days

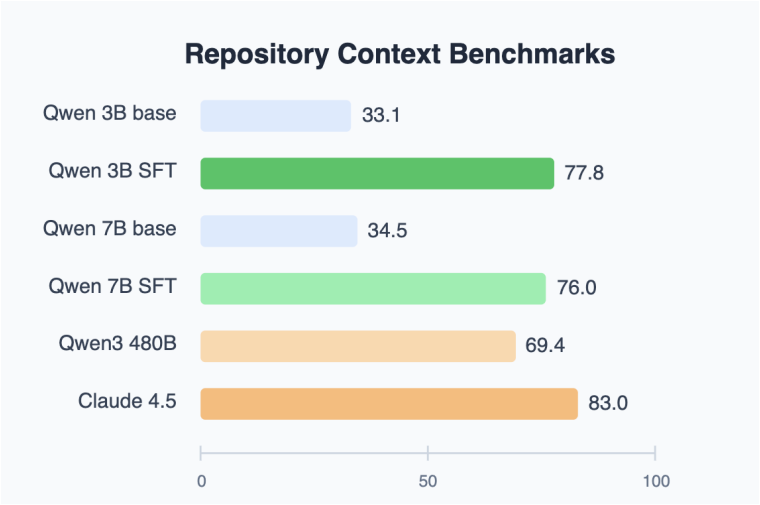
How did you validate?

- Syntax Aware Benchmarks
- Repository-Level Aware Benchmaks

Does it understand the Syntax ?



Does it understand the Context ?

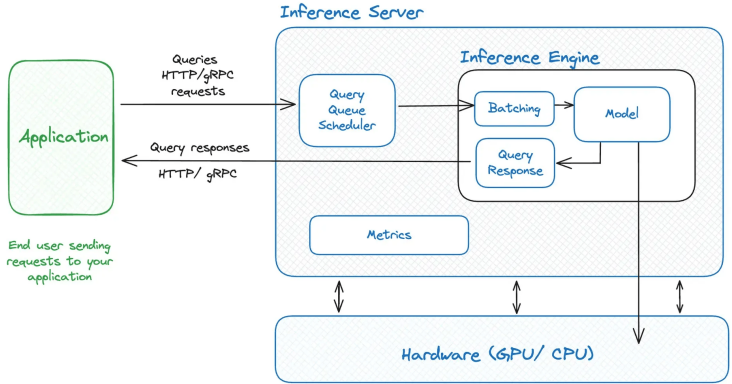


How can we have a sovereign LLM ecosystem in Pharo?

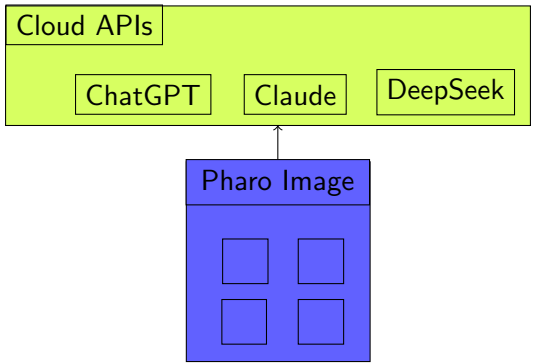
- Having the tools inside Pharo to create, run, and interact with LLMs.
- Having the ability to train and fine-tune our own models.
- Having the ability to deploy and maintain our own models.

How to Deploy Our Models?

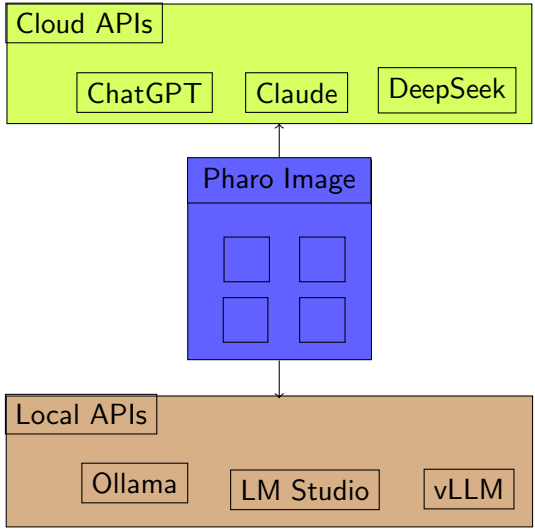
How we can deploy an LLM?



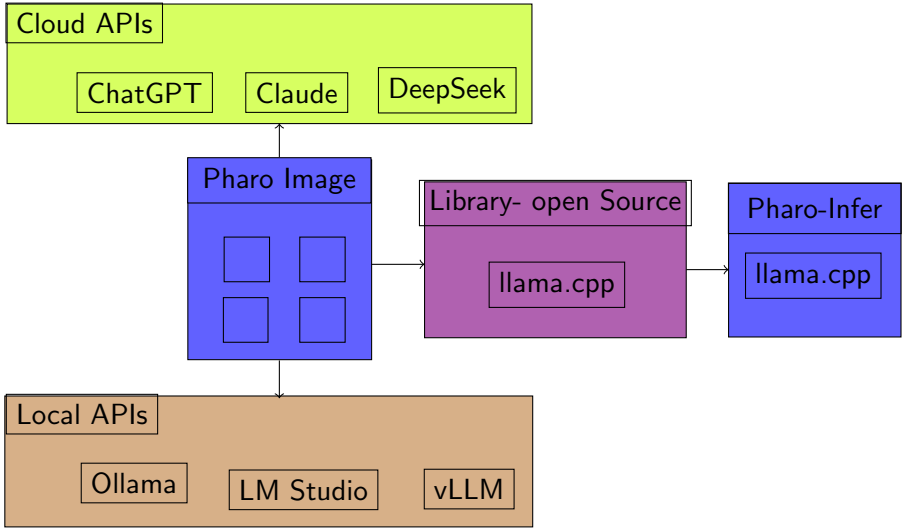
Pharo-LLM Inference Architecture



Pharo-LLM Inference Architecture



Pharo-LLM Inference Architecture



What is Pharo-Infer?

What is an Inference engine?

- It loads a trained language model and prepares it for execution.
- It converts user prompts into tokens and generates tokens back as output.
- It manages memory, context, sampling, and streaming during generation.

Why owning the model matters?

- No artificial limits imposed by someone else.
- Privacy: sensitive data stays on the machine
- Reliability: no network dependency during demos or development.
- Custom behavior for specific domains and workflows.
- Fully integrated in Pharo.

Why not depend on Ollama and LMStudio?

- We inherit their architecture and limitations
- We lose part of the « everything is moldable » inside Pharo.
- Changes in their politics $\longrightarrow$ Direct effect on our infrastructure.
- Building our own engine aligns the full stack with our language and values.

Why do we need our own inference engine?

- Cost
- Rate Limits
- Privacy Concerns
- Network Dependency
- Lack of Control

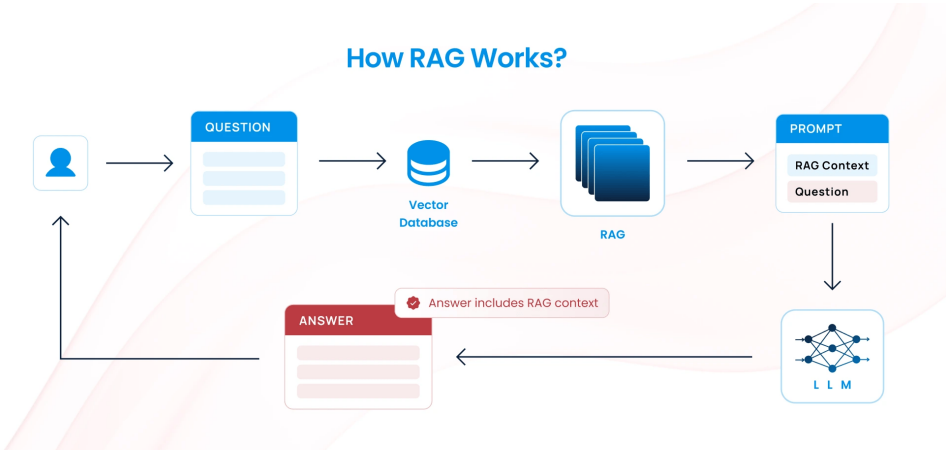
Pros and Cons of our own Inference Engine

Pros	Cons
Independence	More maintenance work
Better privacy	Requires performance optimization
Full integration	Needs hardware and model management

What if I'm working in a team?

- Deploy the model once on a shared server for the whole team.
- Connect it to the team's repositories, documentation, and project knowledge using RAG.
- Keep the model up to date with the latest project changes without retraining it.
- Keep source code and sensitive data inside the organization's infrastructure.
- Share the same APIs infrastructure across the team.

What is a RAG?


How RAG Works?
QUESTION
Vector Database
RAG
PROMPT
RAG Context
Question
L L M
Answer includes RAG context
ANSWER

Pharo-LLM Vision

External Providers	Pharo-LLM
Cost: Paid	Cost: Free
Rate Limits: Serious	Rate Limits: No rate limits
Run on the cloud: No privacy	Run locally
Lack of control	Big control

Pharo-LLM Vision Limitations?

- Training and fine-tuning require time, data preparation, and experimentation.
- Bigger models need more resources and are harder to maintain.
- The community must share datasets, benchmarks, and model improvements.

Conclusion?

- The goal is not to train a huge general-purpose models.
- A 7B model quantized to 4-bit uses around 4.3 GB of memory.
- The benchmarks show that Pharo-specialized models outperform their base versions.
- Fine-tuned models produce more valid Pharo completions.
- Smaller specialized models can outperform much larger general-purpose models on realistic Pharo tasks.
- This supports the idea of a local, sovereign, Pharo-native LLM ecosystem.
- A small model well trained and fine tuned better than a big model trained and fine tuned for general purpose

huggingface.co/pharo-llm

Hugging Face

ollama.com/pharo-llm

Ollama

github.com/pharo-llm

GitHub

arxiv.org

Paper

Pharo-LLM is not only a technical project;
It is a strategic investment in independence
Contributions are very welcome!