Published in Towards AI

Jesus Rodriguez Follow
May 4 · 5 min read · ✨ · Listen

Inside Lamini: A New Framework for Fine-Tuning LLMs

The framework streamlines the process of using techniques such as RLHF in your LLM models.

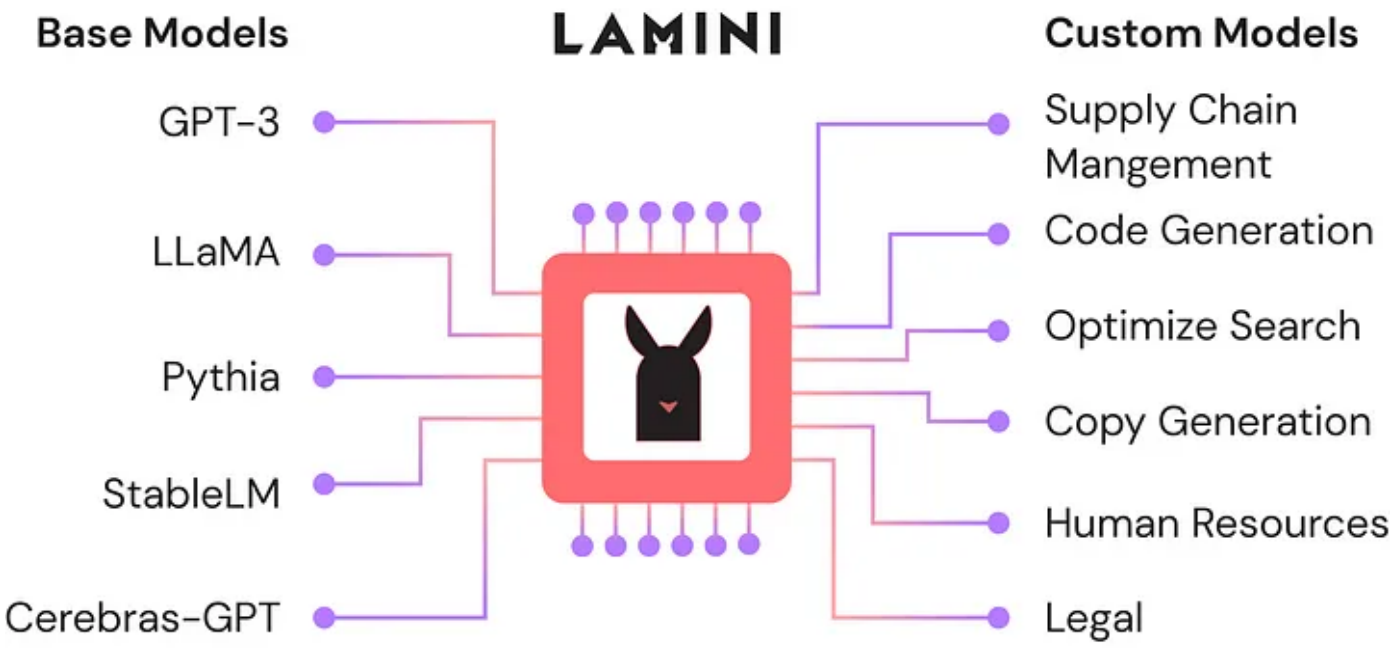

Image Credit: Lamini

> *I recently started an AI-focused educational newsletter, that already has over 150,000 subscribers. TheSequence is a no-BS (meaning no hype, no news etc) ML-oriented newsletter that takes 5 minutes to read. The goal is to keep you up to date with machine learning projects, research papers and concepts. Please give it a try by subscribing below:*

Fine-tuning remains one of the most difficult aspects of the lifecycle of large language models(LLMs) development. This process is particularly challenging if we are talking about techniques such as reinforcement learning with human feedback(RLHF), which require a particularly complicated workflow. Recently, we have seen a new generation of open-source initiatives that attempt to streamline the fine-tuning process in LLMs. One of the most recent additions to that stack is Lamini.

Lamini is a powerful tool that allows developers of all backgrounds to train high-performing LLMs, including models as good as ChatGPT, using just a few lines of code from the Lamini library. The library contains optimizations that go beyond what's currently available to developers, from complex RLHF to simple hallucination reduction. It also makes it easy to compare multiple base models with just one line of code, whether they're from OpenAI or open-source models on HuggingFace.

Lamini includes some key capabilities:

· *The Lamini library includes optimized prompt-tuning and typed outputs, which you can try out in our playground right now.*

· *With only a few lines of code, you can access the advanced Lamini library for fine-tuning and RLHF by signing up for early access.*

· *The hosted data generator enables the building blocks for creating data necessary to train instruction-following LLMs.*

· *An instruction-following LLM that can be used with few lines of code.*

The process of using Lamini can be summarized in the following workflow.

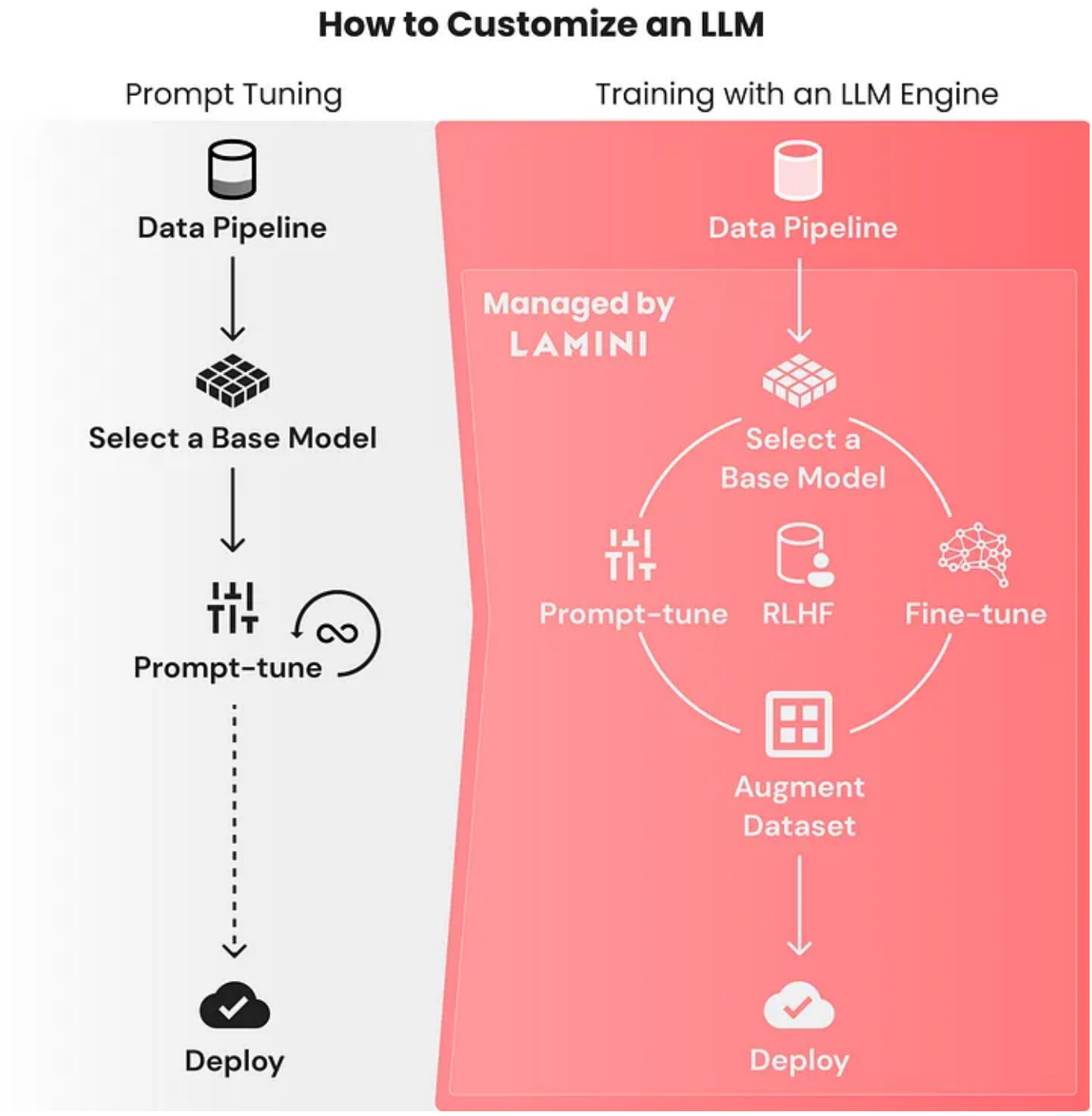

Image Credit: Lamini

1. **Prompt-tune models with ease:** The Lamini library provides APIs that enable developers to prompt-tune models easily, including ChatGPT and other models. The library's APIs support prompt-tuning across different models, allowing developers to swap between OpenAI and open-source models with a single line of code. The Lamini library optimizes the prompt to enable developers to take advantage of different

output pairs is crucial to training a model to respond to its inputs. The dataset helps the model learn how to follow instructions given in English or respond in JSON. Lamini has released a repository that generates 50k data points from as few as 100 data points using the Lamini library, hitting the Lamini engine. This repository includes an open-source 50k dataset. Details on how developers can generate their datasets are available below.

3. **Finetune models on a large dataset:** In addition to the data generator, Lamini has released an LLM that is fine-tuned on the generated data using the Lamini library. Developers can fine-tune their models programmatically with early access to this functionality. Alternatively, developers can start with OpenAI's fine-tuning API.

4. **Run RLHF:** Lamini makes it easy for developers to run RLHF on fine-tuned models without the need for a large team of machine learning and human labeling experts.

5. **Deploy models to the cloud:** Once a developer has fine-tuned their model, deploying it to the cloud is straightforward. Developers can hit the API endpoint in their product or feature to deploy their model.

Using Lamini

Lamini provides a simple programming model for fine tuning models as illustrated in the following code:

```
class Animal(Type):
    name: str = Context("name of the animal")
    n_legs: int = Context("number of legs that animal has")

class Speed(Type):
```

```
    speed: float = Context("how fast something can run")

llama_animal = Animal(name="Larry", n_legs=4)
centipede_animal = Animal(name="Cici", n_legs=100)

my_data = [llama_animal, centipede_animal]

dog_animal = Animal(name="Nacho", n_legs=4)
dog_speed = Story(story="There once was a cute doggo named Nacho. She was a golden

my_data.append([dog_animal, dog_speed])
```


Other interesting capabilities include batching, which enables executing a fine-tuning job as a batch.

```
job = llm.submit_job(self, input, output_type, *args, **kwargs)
```

Additionally, Lamini allows creating add variations to the outputs.

```
ad_copy = llm(input=aspects, output_type=AdCopy, random=True)
```

Or remove duplicates.

```
ad_copies = llm.sample(input=aspects, output_type=AdCopy, n=5)
```

The Data Generator

One of the main components of the Lamini architecture is the Lamini data generator which is a powerful pipeline of LLMs designed to enhance the performance of your LLM using a small set of 100+ instructions paired with their expected responses,

generating over 50k new pairs of instructions and responses inspired by Stanford Alpaca.

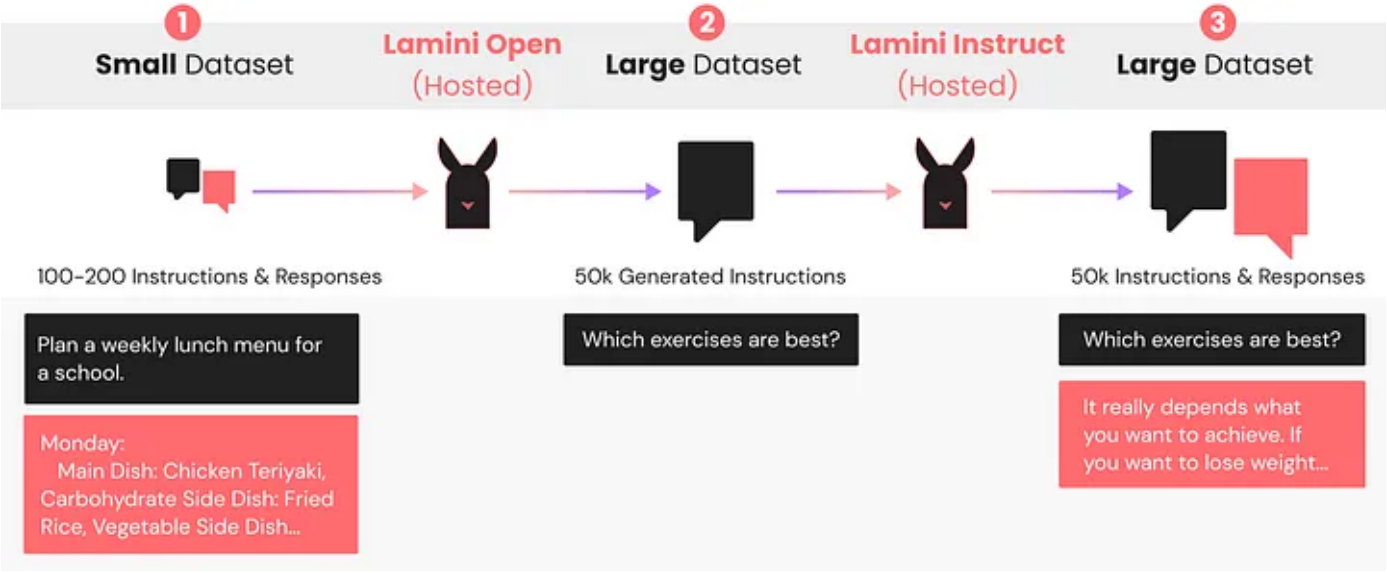

Image Credit: Lamini

This pipeline leverages the Lamini library to call upon different yet similar LLMs to generate diverse pairs of instructions and responses to train your LLM to follow instructions better.

Lamini provides defaults for the generation pipeline using open-source LLMs called Lamini Open and Lamini Instruct. As of this release, the framework is using EleutherAI's Pythia for Lamini Open, which generates more instructions, and Databricks' Dolly for Lamini Instruct, which generates paired responses to those instructions.

Swapping LLMs using Lamini can be done in a few lines of code.

Lamini is tackling one of the most difficult challenges in LLM-driven development. The framework provides a simple and consistent programming models to abstract the fine tuning process across different LLMs. We are likely to see Lamini incorporated into different LLM frameworks in the near future.

Generative Ai Large Language Models Thesequence Artificial Intelligence