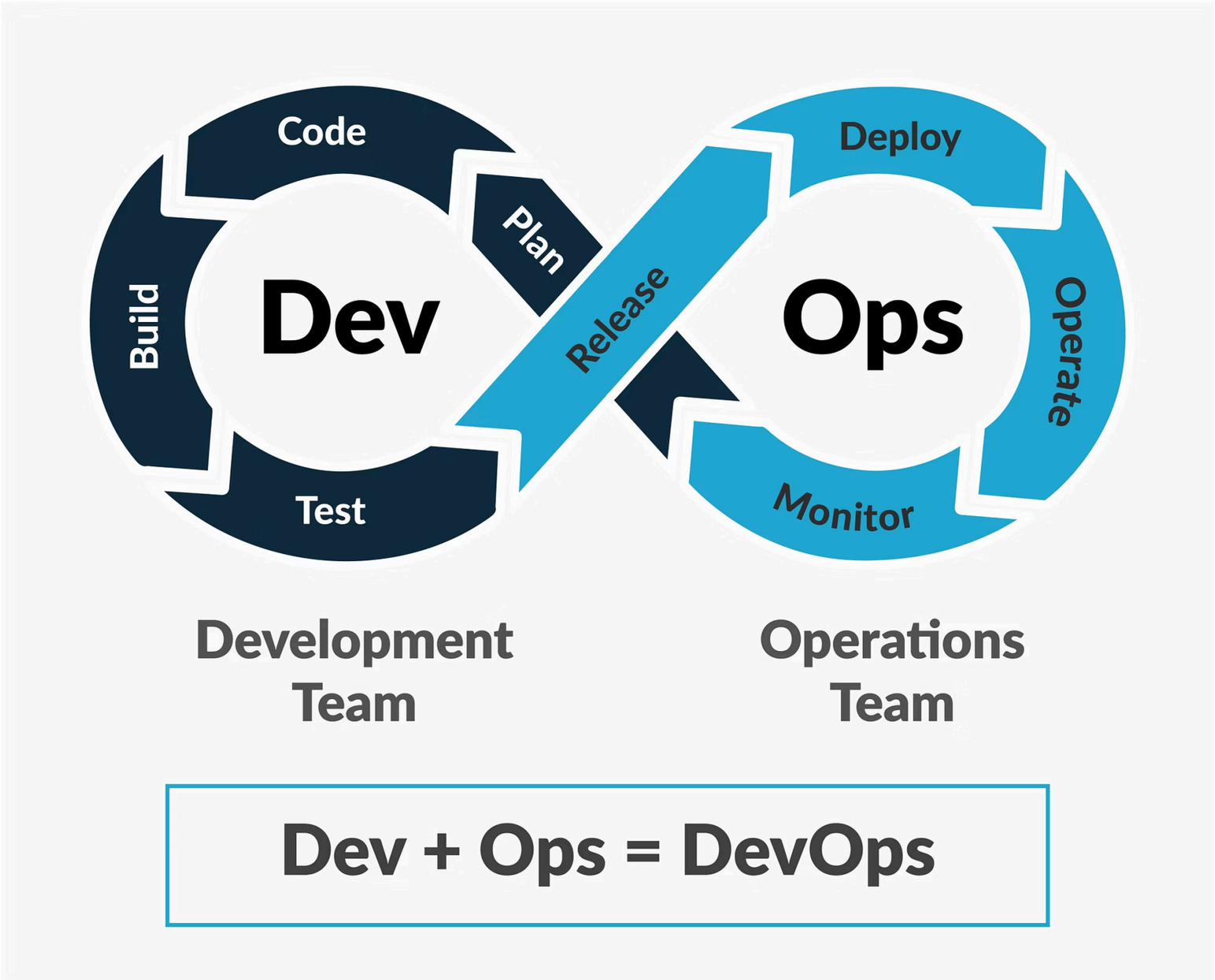
LLM Operations

Andrés Muñoz / Iván Ruiz

Contents

Introduction

DevOps


Code
Plan
Build
Dev
Test
Release
Deploy
Operate
Ops
Monitor
Development Team
Operations Team
Dev + Ops = DevOps

LLM Operations

Set of practices to manage the lifecycle of language models.

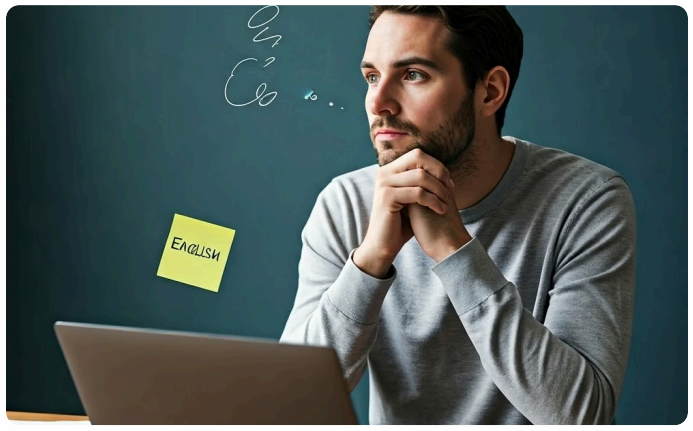
LLM Operations ensures a robust and efficient process for developing, deploying and monitoring these models.

From model training and deployment to performance monitoring and update management, LLM Operations covers all aspects of the language model lifecycle.

It requires close collaboration between data scientists, software developers and system administrators.

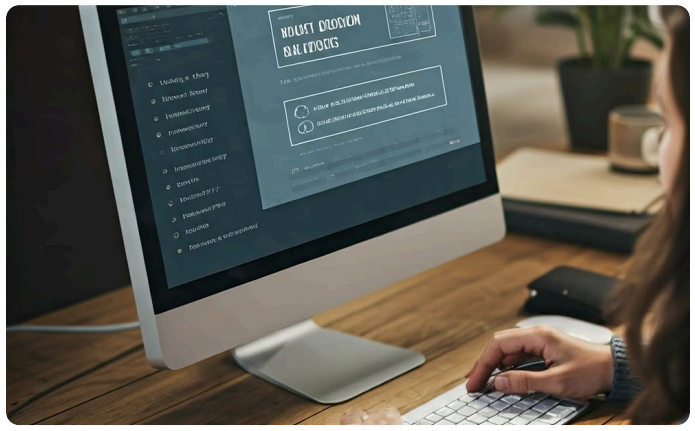
Model Preparation

Model Selection



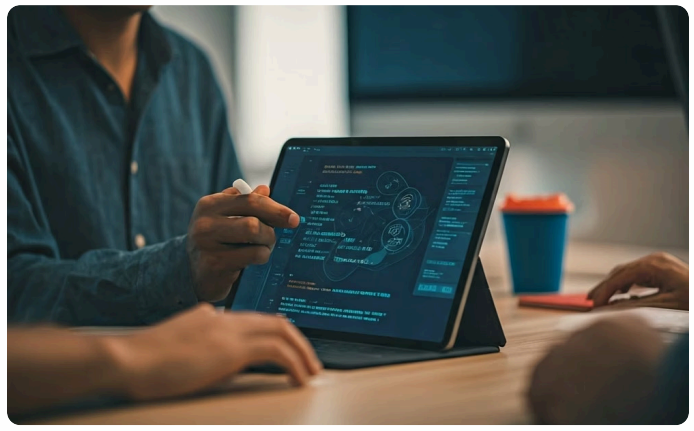
Use Case and Language

First, identify the specific task for which you will use the language model and the desired language.



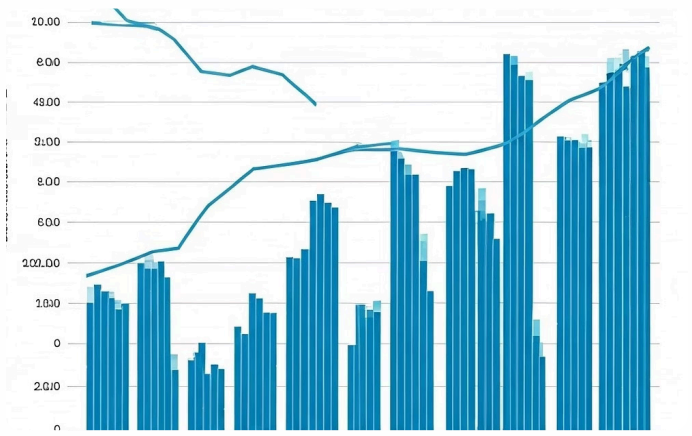
Accessibility and Licensing

Check the availability of the model and the terms of its licence.



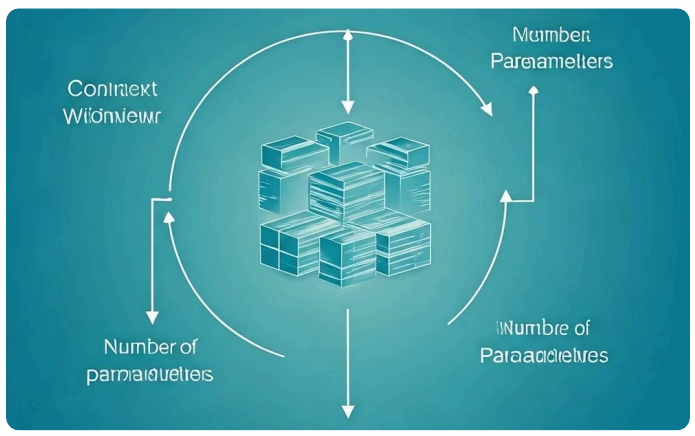
Support and Ecosystem

Consider the availability of support, documentation and community engagement.



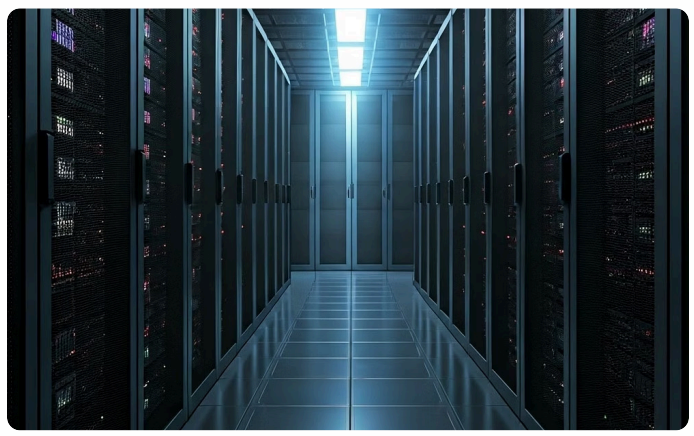
Performance

Check benchmarks and your own evaluations to assess the quality of the model's responses.



Size

Consider the model's context window and number of parameters.



Costs and Infrastructure

Consider the associated costs, including cloud expenses and/or required computing power.

Model Deployment

Cloud

The major cloud service providers offer language models as a service, along with other machine learning models such as speech recognition and synthesis.

On-premises

Local inference engines, such as Ollama, ensure that data remains within our data centres. However, they require a significant hardware infrastructure.

Inference Engines in the Cloud

Google Cloud

Google Cloud Vertex AI

Prueba Vertex AI, una plataforma de desarrollo de IA completamente administrada para compilar apps de IA generativa, con acceso a más de 130...

Amazon Web Services, Inc.

Amazon Bedrock – AWS

La forma más sencilla de crear y escalar aplicaciones de IA generativa con modelos fundacionales

azure.microsoft.com

Microsoft Azure AI Services

Amplíe su cobertura con el uso de servicios de inteligencia artificial. Las herramientas y los servicios de inteligencia artificial le ayudan a automatizar el...

On-premises inference engines

GitHub

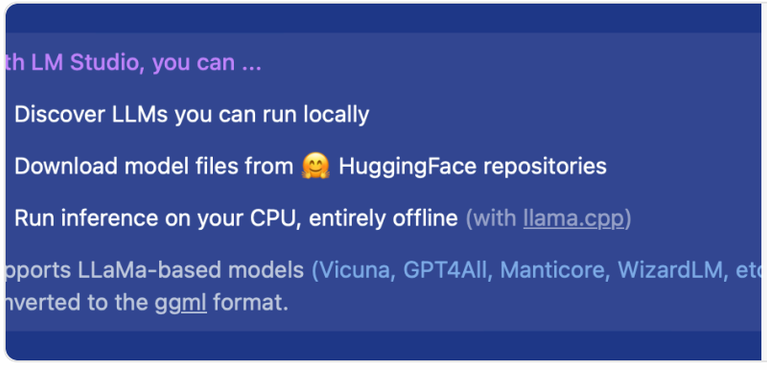
GitHub – ggerganov/llama.cpp: LLM inference in C/C++

LLM inference in C/C++. Contribute to ggerganov/llama.cpp development by creating an account on GitHub.

localai.io

Overview

What is LocalAI?



lmstudio.ai

👾 LM Studio – Discover and run LLMs locally

Find, download, and experiment with LLMs on your locally on your laptop

Ex 1. Model Preparation

Llama

Get up and running with large language models.

Run Llama 3.2, Phi 3, Mistral, Gemma 2, and other models. Customize and create your own.

Download ↓

Available for macOS, Linux, and Windows (preview)

Ollama Installation

Download the Docker script (linux)

```
curl -fsSL https://ollama.com/install.sh | sh
```

Configure the service

```
[Unit]
Description=Ollama Service
After=network-online.target

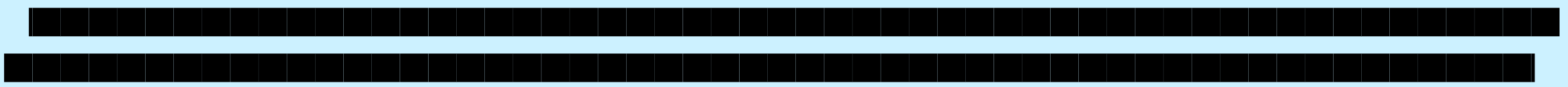
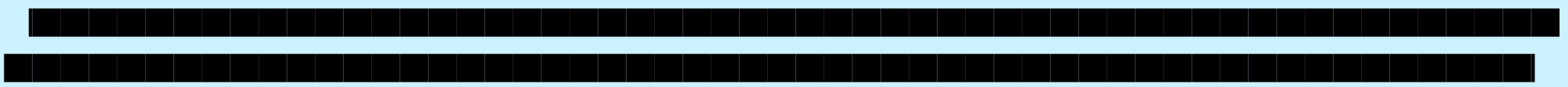
[Service]
ExecStart=/usr/local/bin/ollama serve
User=user
Group=user
Restart=always
RestartSec=3
Environment="PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin"
Environment="OLLAMA_MODELS=/home/user/llm"
Environment="OLLAMA_HOST=0.0.0.0"

[Install]
WantedBy=default.target
```

Download model

```
user@server:~$ ollama pull llama3.2
pulling manifest
pulling dde5aa3fc5ff... 100%

2.0 GB
pulling 966de95ca8a6... 100%

1.4 KB
pulling fcc5a6bec9da... 100%

7.7 KB
pulling a70ff7e570d9... 100%

6.0 KB
verifying sha256 digest
writing manifest
```

Check installed models

```
user@server:~$ ollama list
NAME              ID              SIZE      MODIFIED
llama3.2:latest    a80c4f17acd5    2.0 GB    5 minutes ago
phi3:3.8b          4f2222927938    2.2 GB    29 hours ago
```

Load model into memory and use from the CLI

```
user@server:~$ ollama run llama3.2
>>> hello
Hello! How can I assist you today?
>>> Send a message (/? for help)
```

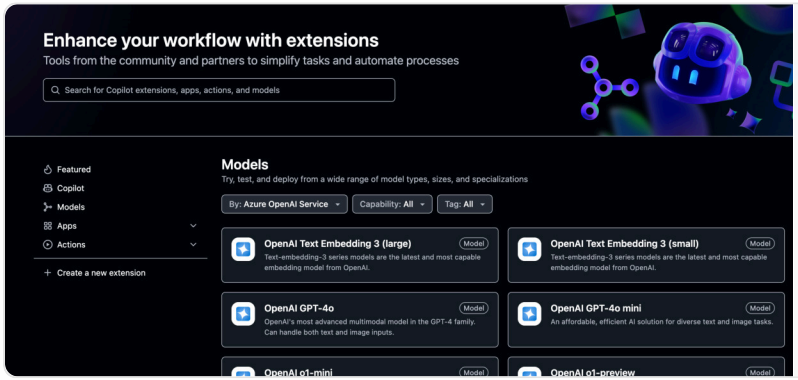
Stop model

```
ollama stop llama3.2
```

Github models

Configuring Github models

Access with the browser to https://github.com/marketplace/models



GitHub

GitHub

GitHub is where people build software. More than 100 million people use GitHub to discover, fork, and contribute to over 420 million projects.

Create a personal token from https://github.com/settings/tokens

Development with the model

Model Configuration

Max Tokens

This defines the maximum number of tokens the model can generate during completion. It helps control the length of the generated output.

Temperature

Temperature determines the randomness of the model. Higher temperatures lead to more creative outputs, while lower temperatures are used for fact-based tasks.

Top P / Top K

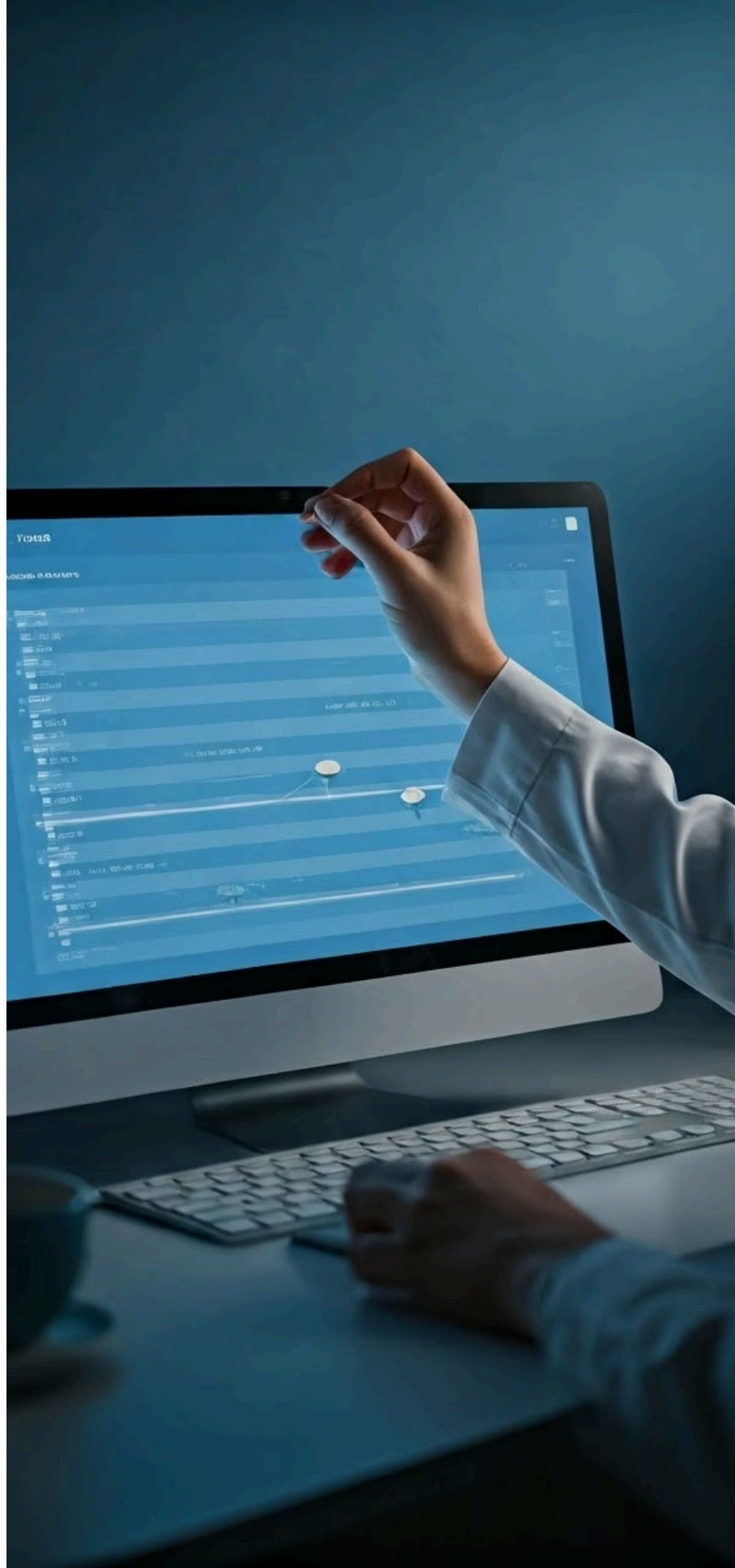
Limits the set of candidate tokens the model can choose from during each generation step. It includes the K most probable words or the necessary ones until the sum of their probabilities reaches a predefined threshold P.

Stop Tokens

These tokens or sequences indicate to the model to stop generating text once they are encountered. It's useful for controlling the flow of the conversation.

JSON Mode

When enabled, all LLM responses are formatted in JSON, enabling structured data processing.



Prompt Engineering

Prompt engineering is crucial for creating instructions that guide the LLM to behave as desired.

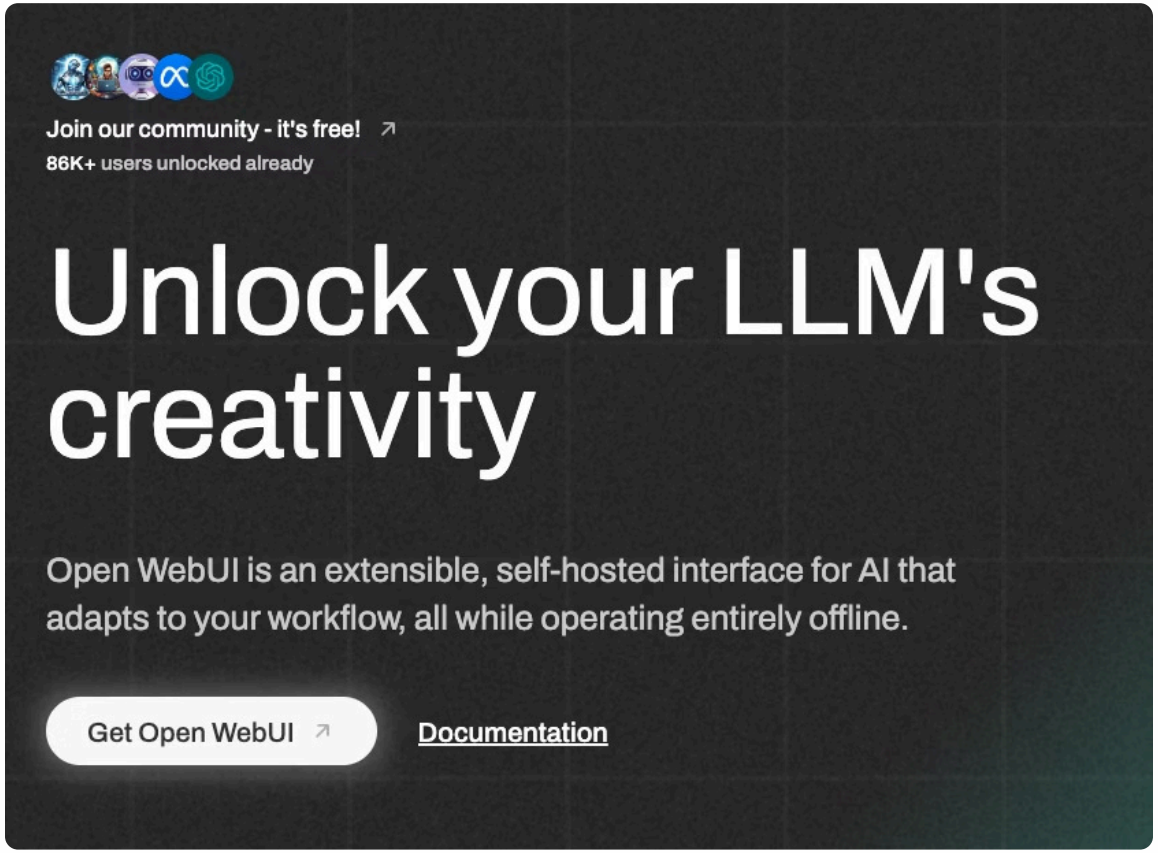
A good prompt must be clear, concise and specific, providing sufficient context to the model to generate a useful response.

To obtain optimal results, it is essential to experiment with different prompts, analyse the results and adjust the prompt until the desired response is obtained.

Prompt engineering becomes an iterative process of trial and error, seeking the best ways to formulate questions and instructions to the LLM.

Ex 2. Use of prompts

Open WebUI

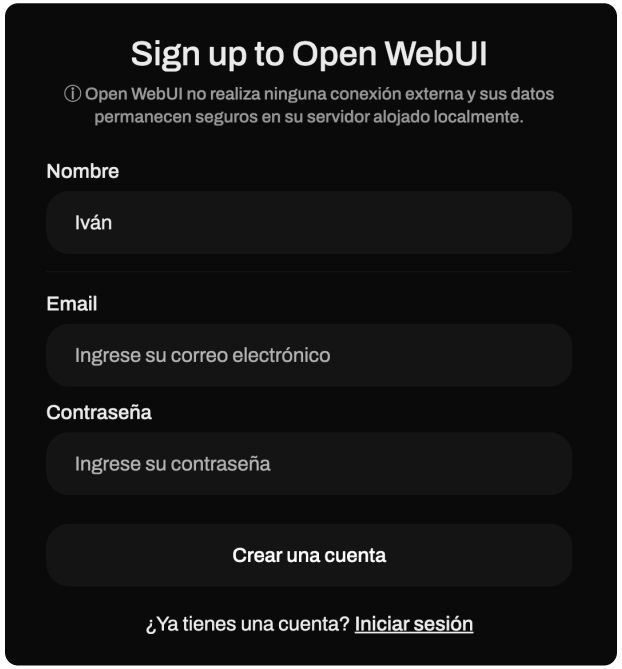

Join our community - it's free!
86K+ users unlocked already
Unlock your LLM's creativity
Open WebUI is an extensible, self-hosted interface for AI that adapts to your workflow, all while operating entirely offline.
Get Open WebUI
Documentation

Installing OpenWebUI

Download and start the Docker container (linux)

```
docker run -d --network=host -v open-webui:/app/backend/data -e OLLAMA_BASE_URL=http://127.0.0.1:11434 --name open-webui --restart always ghcr.io/open-webui/open-webui:main
```

Open a browser and go to http://SERVER:8080 to create a new account

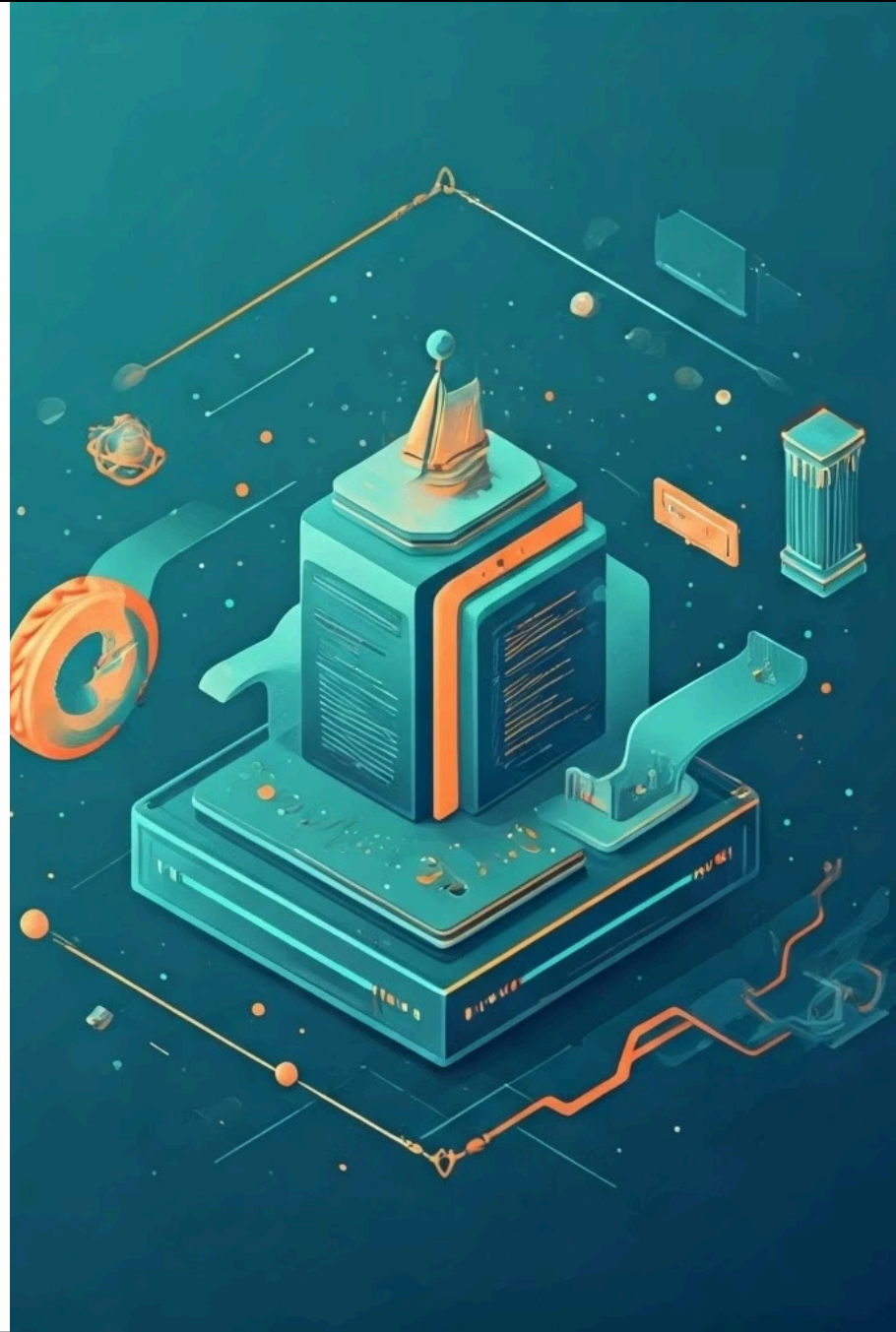


Integrating the model into applications

At this stage, the selected LLM is integrated into applications to enhance their functionality.

To simplify the process, it is recommended to use AI integration frameworks.

Additionally, these frameworks help connect the LLM with other components, such as databases, APIs, and other tools, facilitating the creation of more complex AI applications.



RAG

This technique combines the power of language models with the ability to retrieve information from external sources.

Essentially, RAG allows LLMs to access databases, documents, or websites to obtain relevant information and use it to generate more accurate and informative responses.

Software Testing

Unit Testing

Traditional testing frameworks, such as xUnit, can be adapted to assess the reliability of LLMs. However, the non-deterministic nature of LLMs requires new strategies to measure software confidence.

Dataset Creation

For testing, datasets (input-output pairs) are used, created manually or synthetically generated.

Regression Testing

It is crucial to perform regression testing after implementing different prompting practices, incorporating RAG techniques or changing the LLM.

Ex 3. Unit testing with models

Model Observability

Observability: Traces

Importance in AI systems

Observability allows understanding the internal workings of a system through external monitoring.

Observability is crucial for distributed systems and applications using LLMs, especially when handling requests, chains, RAG and external tools.

Tracing details

Traces facilitate debugging and troubleshooting by identifying the root cause of issues.

Each trace reflects a specific operation or request, detailing its flow through the system.

Additionally, they often include metadata such as user information, session details and tags.

Observability: Qualitative and Quantitative Assessments

Manual Labelling

Manually label traces and observations to evaluate the quality of responses and actions generated by the LLM.

User Feedback Collection

Capture user opinions through mechanisms like thumbs up/down, stars (1-5) or acceptance/rejection of responses.

LLM as a Judge

Use another LLM to evaluate the responses of the target LLM.

Observability: Metrics

Performance

Model usage: Amount of resources used by the model

Latency: Time taken by the model to respond to a request

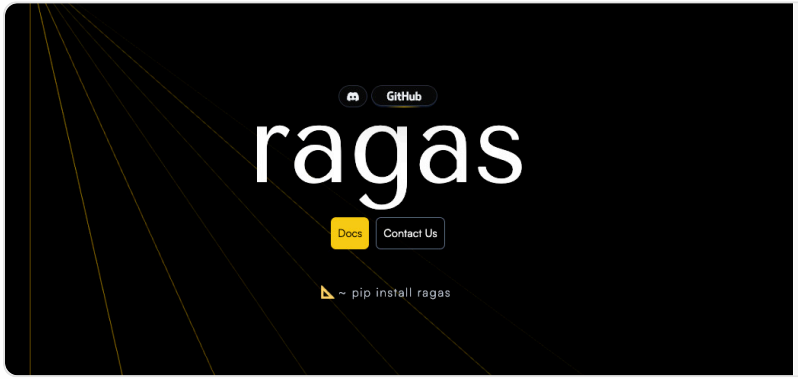
Associated cost: Price associated with using the model

Quality

Precision: Percentage of relevant responses generated by the model

Recall: Percentage of relevant results that are retrieved

Frameworks for testing and evaluation



www.ragas.io

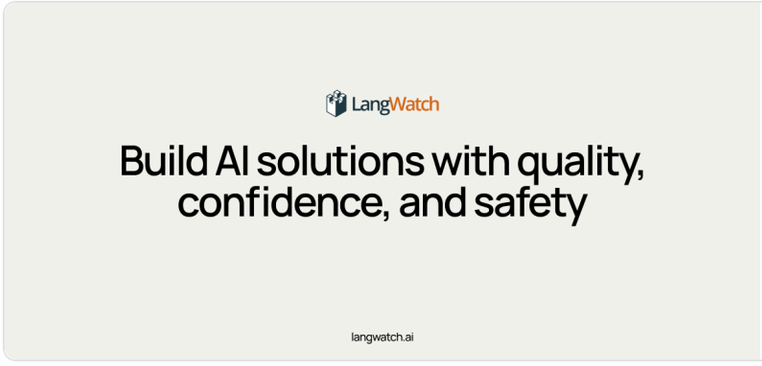
Ragas

Ragas is an open source framework for testing and evaluating LLM applications. Ragas provides metrics , synthetic test data generation and workflows for...

docs.confident-ai.com

DeepEval – The Open-Source LLM Evaluation Framework

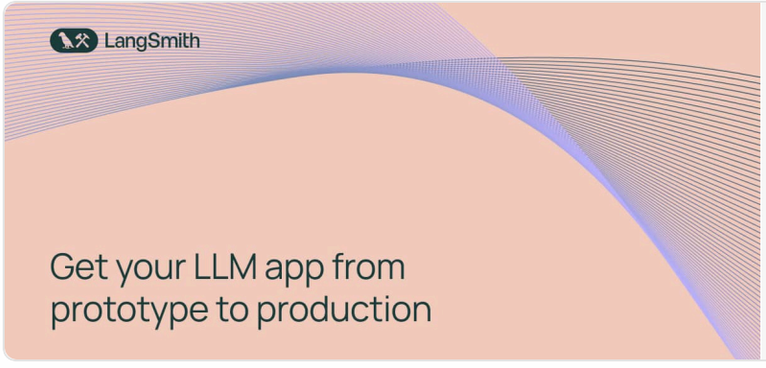
Platforms for LLM Engineering



langwatch.ai

LangWatch – Quality Control & User-Analytics for Generative AI soluti...

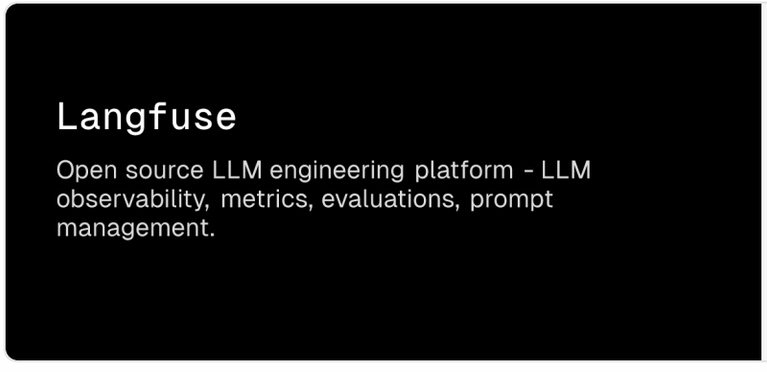
Build AI with confidence. LangWatch safeguards your business from potential AI risks, such as hallucinations and misbehaving users.



www.langchain.com

LangSmith

Get your LLM app from prototype to production.

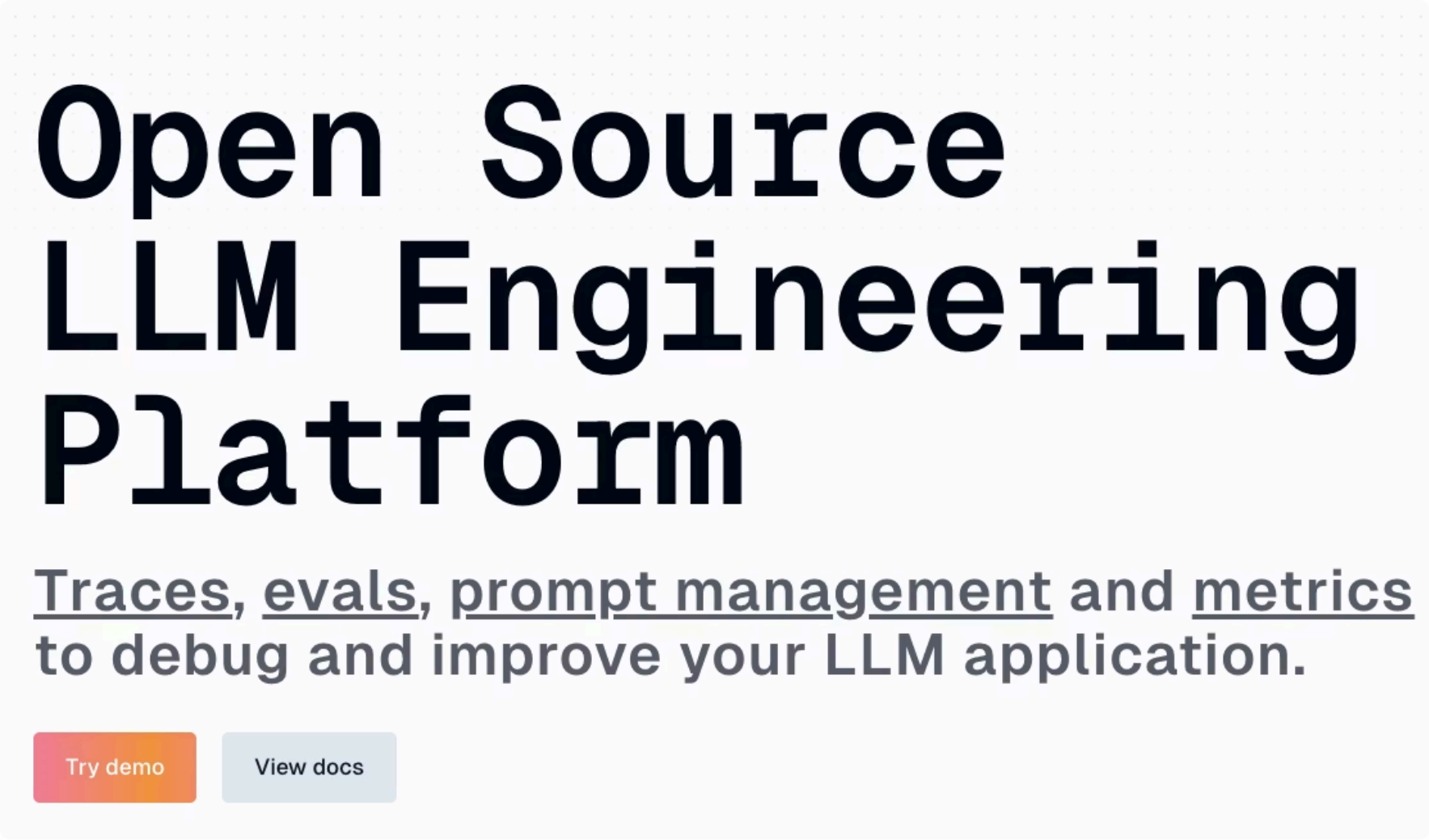


langfuse.com

Langfuse

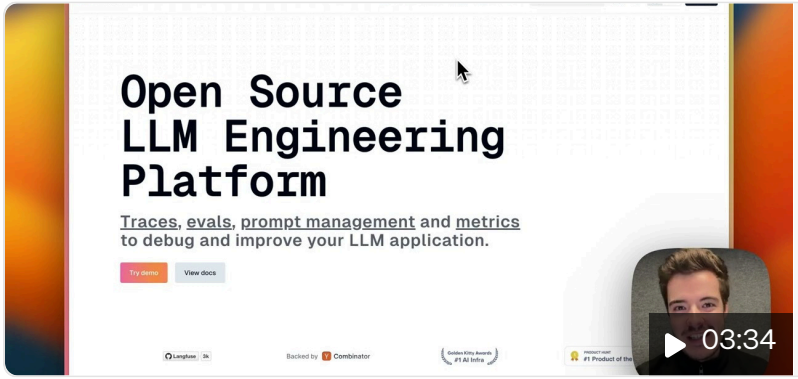
Open source LLM engineering platform – LLM observability, metrics, evaluations, prompt management.

Langfuse


Open Source
LLM Engineering
Platform
Traces, evals, prompt management and metrics
to debug and improve your LLM application.
Try demo
View docs

Ex 4. Observing an AI chatbot

Langfuse demo



YouTube

Introducing Langfuse 2.0 – the Open Source LLM Engineering Platform

Model Retraining

Fine tuning

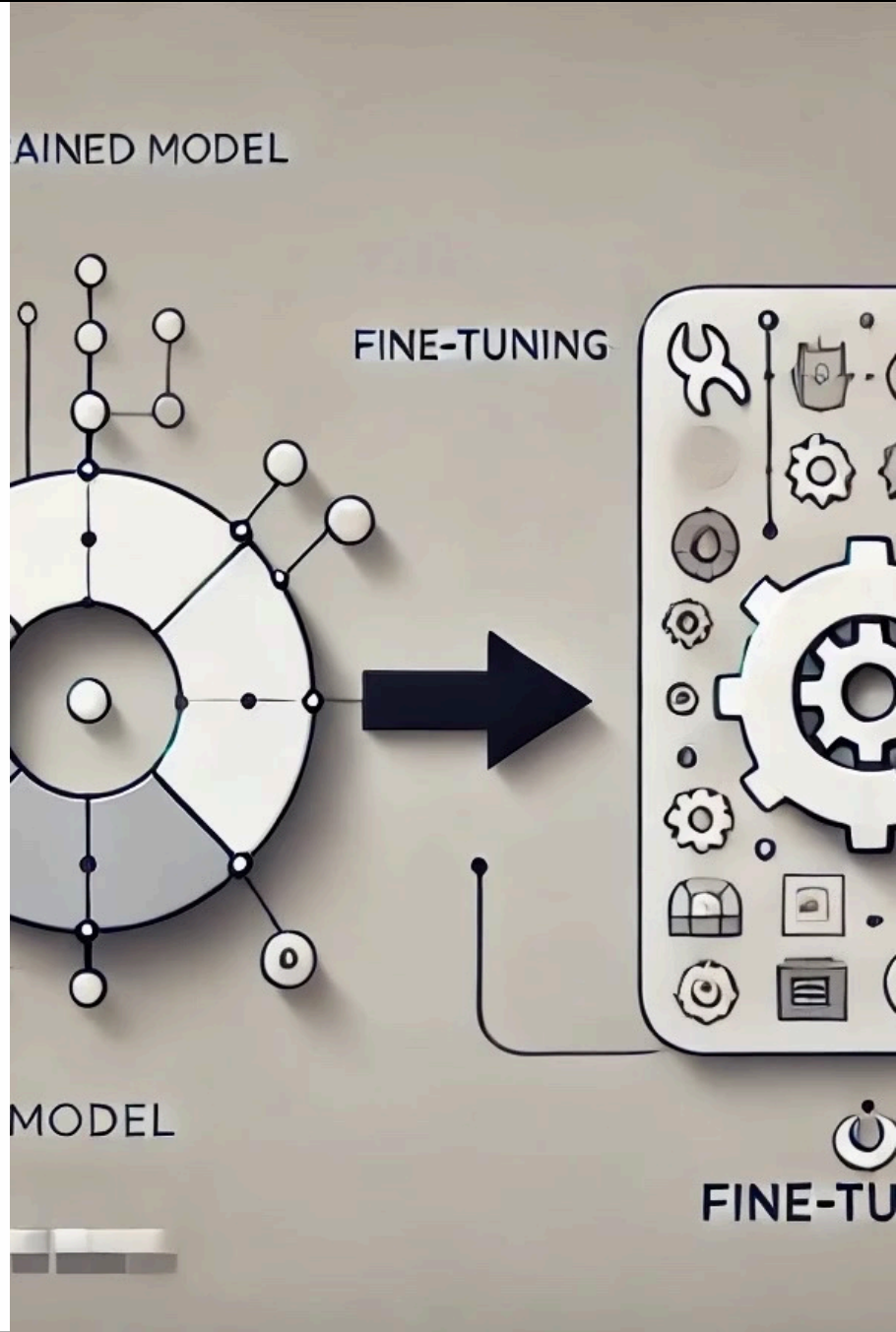
Fine tuning is a technique used to adjust a previously trained model to a new dataset.

This process requires significantly less time and resources compared to training a model from scratch.

It is a valid strategy for scenarios where there is little variability in the data and to optimise the performance of specific tasks.

In scenarios where access to updated and dynamic data is required, it is best to opt for the RAG technique.

One of the potential risks of this technique is that the LLM may accidentally acquire inappropriate knowledge.



Fine-tuning Stages

1 Selection of the base model

A suitable pre-trained model must be chosen for the specific domain.

2 Collect domain-specific data

It is necessary to compile relevant data, such as conversation logs or frequently asked questions, for fine-tuning the model.

3 Data preprocessing

The data must be cleaned and divided into training, test, and validation sets.

4 Hyperparameter definition

Define parameters such as learning rate, batch size, and epochs, as well as the optimizer (e.g. Adam).

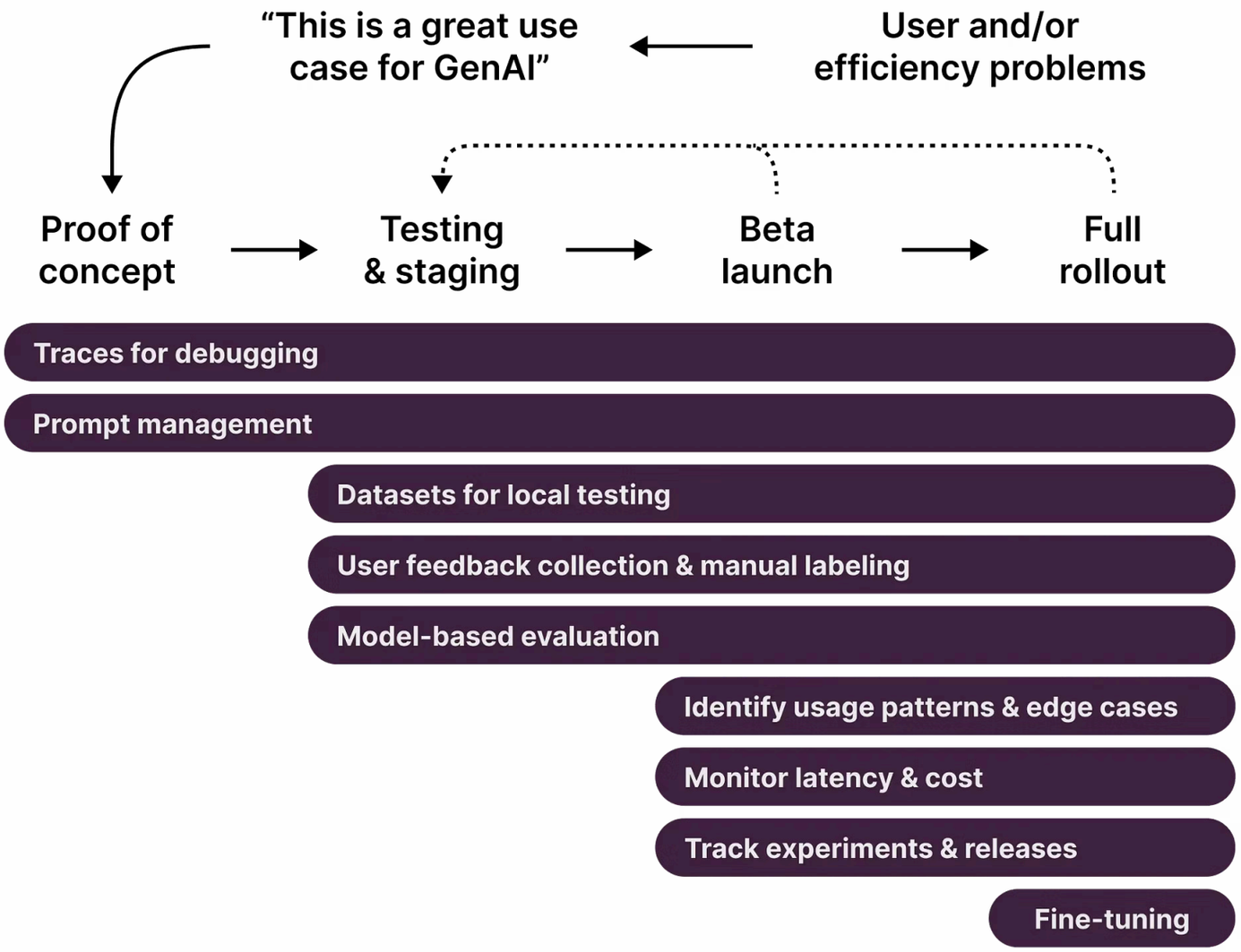
5 Model validation and evaluation

Key metrics, such as accuracy and F1 score, must be used to validate and evaluate the model and avoid overfitting.

6 Model conversion

The model must be converted for use in different environments, for example, exported to ONNX format.

LLM Operations



Summary

This presentation has covered the key aspects of LLM Ops, from model preparation to integration into applications.

The importance of observability and the need for tools to trace the performance, quality and behaviour of models has been highlighted.

We have learned how to use different tools and strategies to monitor and optimise the full lifecycle of language models.