LLM Engine

A software that can load the LLM Models

- [Open WebUI](#)
- [Kuwa Gen AI OS](#)
- [AnythingLLM](#)
- [Ollama](#)
- [LM Studio](#)
- [OpenLLM](#)
- [Bechmark](#)
- [More](#)

Open WebUI

A Web UI Tool for Ollama

URLs

- https://openwebui.com/
- GitHub: https://github.com/open-webui/open-webui
- Docs: https://docs.openwebui.com/

Installation

Installing Both Open WebUI and Ollama Together:

```
# With GPU Support
docker run -d -p 3000:8080 --gpus=all \
  -v ollama:/root/.ollama \
  -v open-webui:/app/backend/data \
  --name open-webui \
  --restart always \
  ghcr.io/open-webui/open-webui:ollama
```

```
# For CPU only
docker run -d -p 3000:8080 \
  -v ollama:/root/.ollama \
  -v open-webui:/app/backend/data \
  --name open-webui \
  --restart always \
  ghcr.io/open-webui/open-webui:ollama
```

Kuwa Gen AI OS

??GenAI???

1. ? ?????GenAI???????????????Windows?Linux
2. ? ?????????? Prompt ?????/??/?????????
3. ? ????? Prompt x RAGs x Bot x ?? x ??/GPUs???????
4. ? ???????????????????????????????????
5. ? ???????????????????????????????

URLs

- [Kuwa AI | Kuwa AI](#)
- [??? | Kuwa AI](#)
- [Kuwa GenAI OS - ?? | Kuwa AI](#)

AnythingLLM

The ultimate AI business intelligence tool. Any LLM, any document, full control, full privacy.

AnythingLLM is a **"single-player"** ??????application you can install on any Mac, Windows, or Linux operating system and get local LLMs, RAG, and Agents with little to zero configuration and full privacy.

> AnythingLLM ?????????????????

You can install AnythingLLM as a Desktop Application, Self Host it locally using Docker and Host it on cloud (aws, google cloud, railway etc..) using Docker

You want AnythingLLM Desktop if...

- You want a one-click installable app to use local LLMs, RAG, and Agents locally
- You do not need multi-user support
- Everything needs to stay only on your device
- You do not need to "publish" anything to the public internet. Eg: Chat widget for website

URLs

- https://useanything.com/
- Doc: https://docs.useanything.com/
- ?????Mintplex-Labs/anything-llm

Ollama

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

- https://ollama.com/
- GitHub: https://github.com/ollama/ollama
- Doc: https://github.com/ollama/ollama/tree/main/docs
- Video: ????????????? AI ?? Ollama ??????????????? - YouTube

Installation

ollama + open webui

```
mkdir ollama-data download open-webui-data
```

docker-compose.yml:

```
services:
  ollama:
    image: ollama/ollama:latest
    ports:
      - 11434:11434
    volumes:
      - ./ollama-data:/root/.ollama
      - ./download:/download
    container_name: ollama
    pull_policy: always
    tty: true
    restart: always
    networks:
      - ollama-docker

  open-webui:
    image: ghcr.io/open-webui/open-webui:main
    container_name: open-webui
    volumes:
```

```
      - ./open-webui-data:/app/backend/data
    depends_on:
      - ollama
    ports:
      - 3000:8080
    environment:
      - 'OLLAMA_BASE_URL=http://ollama:11434'
    extra_hosts:
      - host.docker.internal:host-gateway
    restart: unless-stopped
    networks:
      - ollama-docker


networks:
  ollama-docker:
    external: false
```

ollama

```
mkdir ollama-data download

docker run --name ollama -d --rm \
    -v $PWD/ollama-data:/root/.ollama \
    -v $PWD/download:/download \
    -p 11434:11434 \
    ollama/ollama
```

Models

List Models Installed

```
ollama list
```

Load a GGUF model manually

```
ollama create <my-model-name> -f <modelfile>
```

Page Assist

[Page Assist](#) is an open-source Chrome Extension that provides a Sidebar and Web UI for your Local AI model.

- Video: [This Chrome Extension Surprised Me - YouTube](#)

LM Studio

Discover, download, and run local LLMs.

With LM Studio, you can ...

? - Run LLMs on your laptop, entirely offline
? - Use models through the in-app Chat UI or an OpenAI compatible local server
? - Download any compatible model files from HuggingFace
? repositories
? - Discover new & noteworthy LLMs in the app's home page

URLs

- https://lmstudio.ai/
- Doc: https://lmstudio.ai/docs

OpenLLM

OpenLLM helps developers run any open-source LLMs, such as Llama 2 and Mistral, as OpenAI-compatible API endpoints, locally and in the cloud, optimized for serving throughput and production deployment.

- GitHub: https://github.com/bentoml/OpenLLM
- CoLab: https://colab.research.google.com/github/bentoml/OpenLLM/blob/main/examples/llama2.ipynb

Install

> Recommend using a Python Virtual Environment

```
pip install openllm
```

Start a LLM Server

```
openllm start microsoft/Phi-3-mini-4k-instruct --trust-remote-code
```

To interact with the server, you can visit the web UI at http://localhost:3000/ or send a request using curl. You can also use OpenLLM's built-in Python client to interact with the server:

```
import openllm

client = openllm.HTTPClient('http://localhost:3000')
client.generate('Explain to me the difference between "further" and "farther"')
```

OpenAI Compatible Endpoints

```
import openai

client = openai.OpenAI(base_url='http://localhost:3000/v1', api_key='na')  # Here the server is running on 0.0.0.0:3000

completions = client.chat.completions.create(
```

```
  prompt='Write me a tag line for an ice cream shop.', model=model, max_tokens=64, stream=stream
)
```

LangChain

```
from langchain.llms import OpenLLMAPI

llm = OpenLLMAPI(server_url='http://44.23.123.1:3000')
llm.invoke('What is the difference between a duck and a goose? And why there are so many Goose in Canada?')

# streaming
for it in llm.stream('What is the difference between a duck and a goose? And why there are so many Goose in Canada?'):
  print(it, flush=True, end='')

# async context
await llm.ainvoke('What is the difference between a duck and a goose? And why there are so many Goose in Canada?')

# async streaming
async for it in llm.astream('What is the difference between a duck and a goose? And why there are so many Goose in Canada?'):
  print(it, flush=True, end='')
```

Bechmark

Benchmark for LLM engines

bench.py

- [ollama ??????? vllm ????????????_ollama vllm-CSDN??](#)
- YT: [ollama vs vllm - ??????? ollama ? vllm ?????? - YouTube](#)

```
import aiohttp
import asyncio
import time
from tqdm import tqdm

import random

questions = [
    "Why is the sky blue?", "Why do we dream?", "Why is the ocean salty?", "Why do leaves change color?",
    "Why do birds sing?", "Why do we have seasons?", "Why do stars twinkle?", "Why do we yawn?",
    "Why is the sun hot?", "Why do cats purr?", "Why do dogs bark?", "Why do fish swim?",
    "Why do we have fingerprints?", "Why do we sneeze?", "Why do we have eyebrows?", "Why do we have hair?",
    "Why do we have nails?", "Why do we have teeth?", "Why do we have bones?", "Why do we have muscles?",
    "Why do we have blood?", "Why do we have a heart?", "Why do we have lungs?", "Why do we have a brain?",
    "Why do we have skin?", "Why do we have ears?", "Why do we have eyes?", "Why do we have a nose?",
    "Why do we have a mouth?", "Why do we have a tongue?", "Why do we have a stomach?", "Why do we have intestines?",
    "Why do we have a liver?", "Why do we have kidneys?", "Why do we have a bladder?", "Why do we have a pancreas?",
    "Why do we have a spleen?", "Why do we have a gallbladder?", "Why do we have a thyroid?", "Why do we have adrenal glands?",
    "Why do we have a pituitary gland?", "Why do we have a hypothalamus?", "Why do we have a thymus?", "Why do we have lymph nodes?",
    "Why do we have a spinal cord?", "Why do we have nerves?", "Why do we have a circulatory system?", "Why do we have a respiratory system?",
    "Why do we have a digestive system?", "Why do we have an immune system?"
]
```

```python
async def fetch(session, url):
    """
    □□:
        session (aiohttp.ClientSession): □□□□□□□□
        url (str): □□□□□□ URL□

    □□:
        tuple: □□□□ token □□□□□□□□
    """
    start_time = time.time()

    # □□□□□□□□
    question = random.choice(questions) # <--- □□□□□□□□□

    # □□□□□
    # question = questions[0]        # <--- □□□□□□□□□

    # □□□□□
    json_payload = {
        "model": "llama3:8b-instruct-fp16",
        "messages": [{"role": "user", "content": question}],
        "stream": False,
        "temperature": 0.7 # □□□□ 0.7 □□□□□□□□□□□
    }
    async with session.post(url, json=json_payload) as response:
        response_json = await response.json()
        end_time = time.time()
        request_time = end_time - start_time
        completion_tokens = response_json['usage']['completion_tokens'] # □□□□□□□□□□□□ token □□□□
        return completion_tokens, request_time

async def bound_fetch(sem, session, url, pbar):
    # □□□□□ sem □□□□□□□□□□□□□□□□□□□□□□□□□□□
    async with sem:
        result = await fetch(session, url)
        pbar.update(1)
        return result

async def run(load_url, max_concurrent_requests, total_requests):
```

```python
    """
    □□□□□□□□□□□□□□□□□□

    □□:
        load_url (str): □□□□□□URL□
        max_concurrent_requests (int): □□□□□□□□
        total_requests (int): □□□□□□□□□

    □□:
        tuple: □□□□ token □□□□□□□□□□□□
    """
    # □□ Semaphore □□□□□□□□□□
    sem = asyncio.Semaphore(max_concurrent_requests)

    # □□□□□□□HTTP□□
    async with aiohttp.ClientSession() as session:
        tasks = []

        # □□□□□□□□□□□□□□□
        with tqdm(total=total_requests) as pbar:
            # □□□□□□□□□□□□□
            for _ in range(total_requests):
                # □□□□□□□□□□□□□□□□□□□□
                task = asyncio.ensure_future(bound_fetch(sem, session, load_url, pbar))
                tasks.append(task)  # □□□□□□□□□□□

            # □□□□□□□□□□□□□□□
            results = await asyncio.gather(*tasks)

        # □□□□□□□□□□token□□
        completion_tokens = sum(result[0] for result in results)

        # □□□□□□□□□□□□
        response_times = [result[1] for result in results]

        # □□□□token□□□□□□□□□□□
        return completion_tokens, response_times

if __name__ == '__main__':
    import sys
```

```python
if len(sys.argv) != 3:
    print("Usage: python bench.py <C> <N>")
    sys.exit(1)

C = int(sys.argv[1])  # □□□□□
N = int(sys.argv[2])  # □□□□

# vllm □ ollama □□□□ openai □ api □□□□□□□□□
url = 'http://localhost:11434/v1/chat/completions'

start_time = time.time()
completion_tokens, response_times = asyncio.run(run(url, C, N))
end_time = time.time()

# □□□□□
total_time = end_time - start_time
# □□□□□□□□□□□
avg_time_per_request = sum(response_times) / len(response_times)
# □□□□□□ token □□
tokens_per_second = completion_tokens / total_time

print(f'Performance Results:')
print(f'  Total requests         : {N}')
print(f'  Max concurrent requests   : {C}')
print(f'  Total time              : {total_time:.2f} seconds')
print(f'  Average time per request  : {avg_time_per_request:.2f} seconds')
print(f'  Tokens per second        : {tokens_per_second:.2f}')
```

More

LocalAI

LocalAI is the free, Open Source OpenAI alternative. LocalAI act as a drop-in replacement REST API that's compatible with OpenAI API specifications for local inferencing. It allows you to run LLMs, generate images, audio (and not only) locally or on-prem with consumer grade hardware, supporting multiple model families and architectures.

- [Overview | LocalAI documentation](#)
- GitHub: https://github.com/mudler/LocalAI

OpenAI Proxy

Proxy Server to call 100+ LLMs in a unified interface & track spend, set budgets per virtual key/user

Features:

- **Unified Interface**: Calling 100+ LLMs Huggingface/Bedrock/TogetherAI/etc. in the OpenAI ChatCompletions & Completions format
- **Cost tracking**: Authentication, Spend Tracking & Budgets Virtual Keys
- **Load Balancing**: between Multiple Models + Deployments of the same model - LiteLLM proxy can handle 1.5k+ requests/second during load tests.

> ????? LLM ?? ?? OpenAI Proxy ????????????????
> - ?? API ???????
> - ????
> - ????

- Doc: https://docs.litellm.ai/docs/simple_proxy

Xinference

Xorbits Inference (Xinference) is an open-source platform to streamline the operation and integration of a wide array of AI models. With Xinference, you're empowered to run inference

using any open-source LLMs, embedding models, and multimodal models either in the cloud or on your own premises, and create robust AI-driven applications.

- [Welcome to Xinference! — Xinference](#)
- GitHub: https://github.com/xorbitsai/inference

NVIDIA NIM

Explore the latest community-built AI models with an API optimized and accelerated by NVIDIA, then deploy anywhere with NVIDIA NIM inference microservices.

- [NVIDIA NIM for Deploying Generative AI | NVIDIA](#)
- Doc: [Introduction - NVIDIA Docs](#)
- Models: [google / gemma-7b](#)
- YT: [Self-Host and Deploy Local LLAMA-3 with NIMs - YouTube](#)

text-generation-webui

A Gradio web UI for Large Language Models.

> ???????????????? API?

????????? AI ??

- Chat
- Fine-Tune Model
- Multiple model backends: Transformers, llama.cpp (through llama-cpp-python), ExLlamaV2, AutoGPTQ, AutoAWQ, GPTQ-for-LLaMa, QuIP#.
- OpenAI-compatible API server with Chat and Completions endpoints

??

- GitHub: https://github.com/oobabooga/text-generation-webui
- GitHub: https://github.com/Atinoda/text-generation-webui-docker
- [??????LLMs???? ???? (?) - HackMD](#)
  - YOUTUBE [[?? TextGen](#)]
  - YOUTUBE [[????????](#)]
  - YOUTUBE [[??AI??](#)]
  - YOUTUBE [[????](#)]
  - YOUTUBE [[??????](#)]

- ??? [Z01_TextGen_Colab.ipynb](#)
- ?????????? (account:nchc password:nchc) ?????