

Gen AI

???????

- [LLM Models](#)
- [Voice](#)
- [RAG](#)
- [Fine-Tune](#)
- [AI Applications](#)
- [AI Dev](#)
- [Learning AI](#)
- [RedHat AI](#)
- [Agent](#)
- [AI Cloud Providers](#)
- [Prompt Engineering](#)
- [Function Calling](#)
- [Python Coding](#)
- [LLM Engine](#)
 - [Open WebUI](#)
 - [Kuwa Gen AI OS](#)
 - [AnythingLLM](#)
 - [Ollama](#)
 - [LM Studio](#)
 - [Text generation web UI](#)
 - [OpenLLM](#)
 - [NVIDIA NIM](#)
 - [Xinference](#)
 - [Bechmark](#)
 - [OpenAI Proxy](#)
- [Translator](#)

- [Jupyter Notebook](#)
- [LangChain](#)
- [Finance AI](#)
- [Semantic Kernel](#)

LLM Models

Chinese LLMs

- [Taiwan LLM](#) - Project TAME (TAiwanese Mixture of Experts)
 - GitHub: <https://github.com/MiuLab/Taiwan-LLM>
 - HF: <https://huggingface.co/yentinglin>
 - HF: <https://huggingface.co/audreyt>
 - [????LLM?????Project TAME?5,000??Token????????? | iThome](#)
- [TAIDE](#) - Trustworthy AI Dialogue Engine
 - GitHub: <https://github.com/taide-taiwan>
 - HF: <https://huggingface.co/taide>
 - [TAIDE | iThome](#)
- 01.AI - [Yi](#)
 - GitHub: <https://github.com/01-ai/Yi>
 - HF: <https://huggingface.co/01-ai/>
- CKIP-Llama-2-7b ??????????(CKIP)????????????????????????????????Llama-2-7b??Atom-7b????????????????????????405????????????????????????????70?(7 billion)?
 - GitHub: <https://github.com/f901107/CKIP-Llama-2-7b>
 - HF: <https://huggingface.co/spaces/ckiplab/CKIP-Llama-2-7b-chat>
- [Qwen](#) - ????????
 - GitHub: <https://github.com/QwenLM/Qwen>
 - GitHub: <https://github.com/QwenLM/Qwen2>
 - HF: <https://huggingface.co/Qwen>
 - Doc: <https://help.aliyun.com/zh/dashscope/create-a-chat-foundation-model?spm=a2c4g.11186623.0.0.20ea4937azFCan>
- GLM-4 - ?? AI ????????????
 - GitHub: <https://github.com/THUDM/GLM-4>
 - HF: <https://huggingface.co/collections/THUDM/glm-4-665fcf188c414b03c2f7e3b7>
- [Chinese-Mixtral](#)
- [DeepSeek](#) - ?????
 - GitHub: <https://github.com/deepseek-ai/DeepSeek-V2>
 - HF: <https://huggingface.co/deepseek-ai/DeepSeek-V2>

Code LLMs

- [Granite](#) - Open sourcing IBM's Granite code models
 - H F: <https://huggingface.co/ibm-granite>
 - GitHub: <https://github.com/ibm-granite>
 - [IBM????????Granite???????????????????? | iThome](#)
- [Codestral](#) - Mistral's first generative AI model for code
 - HF: <https://huggingface.co/mistralai/Codestral-22B-v0.1>
 - [Mistral AI???????????????? | iThome](#)

LLM Evaluation

- [PromptBench](#): A Unified Library for Evaluating and Understanding Large Language Models.
- AI?????????: [AI?????????.xlsx](#)

Function Calling LLMs

- [Firefunction-v2](#)
 - HF: <https://huggingface.co/fireworks-ai/firefunction-v2>

Content Safty

- [Google ShieldGemma](#)
ShieldGemma??4????????????????????
????????????????????

??

- [???? Model ???? GPU VRAM](#)

RAG

?????? - Retrieval Augmented Generation

RAG ??????????????LLM????????????????/????hallucination????????RAG
?????????retrieval?????????generation???????????????????????????????????? LLM
????????????????????

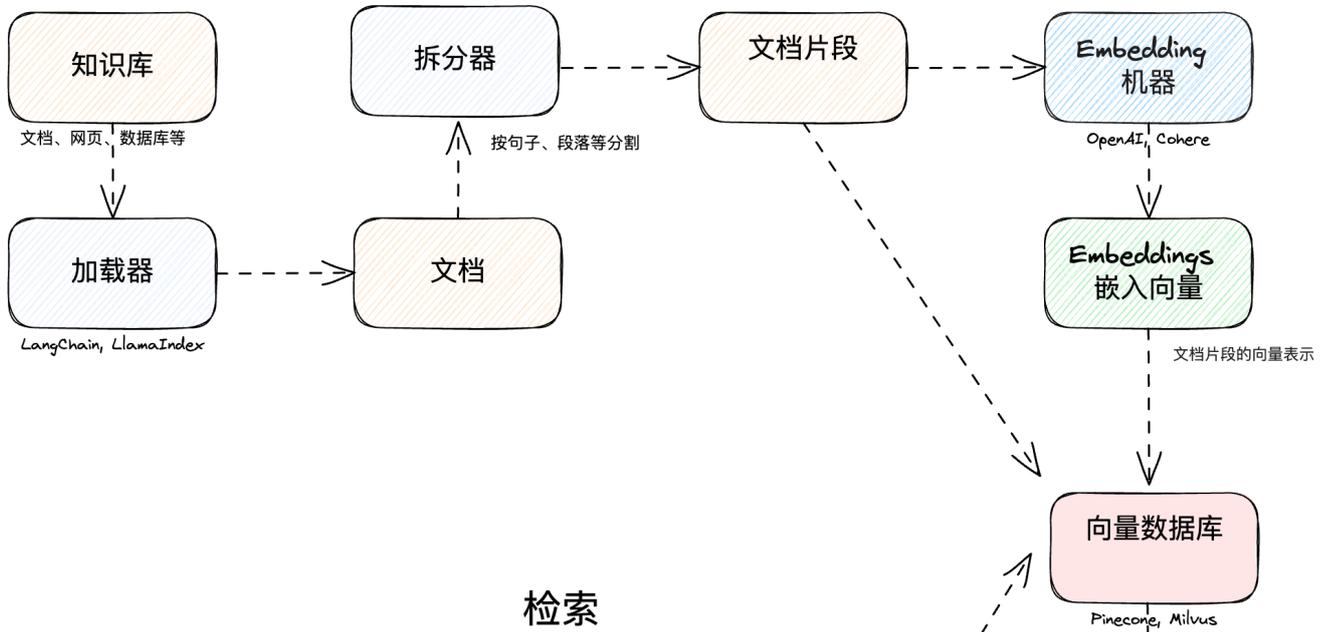
RAG ???

- ?? AI ??
- ?????????
- ???????
- ???????

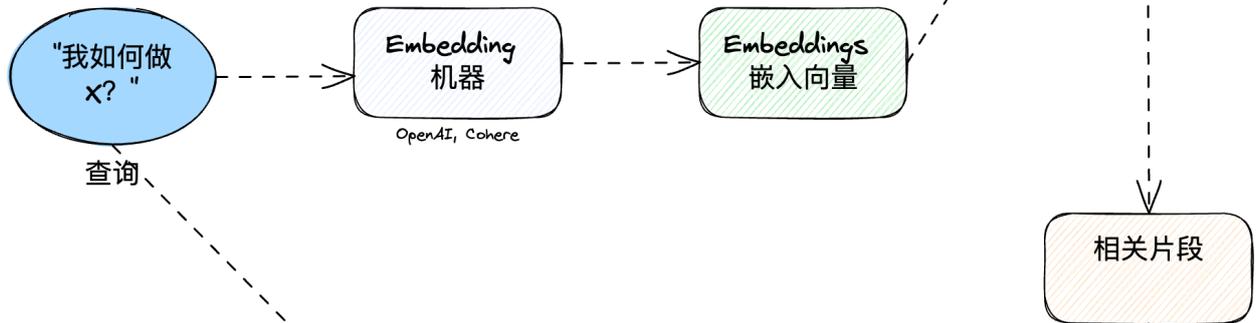
??????

Retrieval-Augmented Generation 检索增强生成

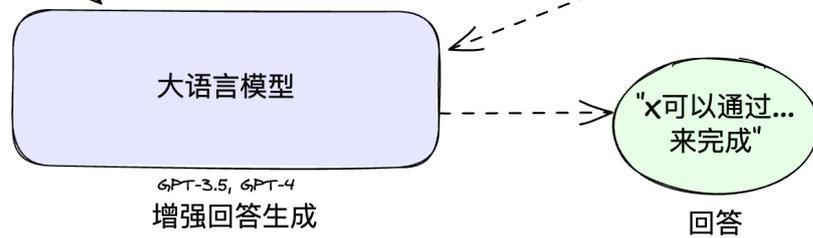
索引



检索



生成



Tutorials

- [ollama + Langchain + Gradio RAG ?????](#)

- [A flexible Q&A-chat-app for your selection of documents with langchain, Streamlit and chatGPT | by syrom | Medium](#)
- [????4?????AI?????????ChatGPT??????|???? BusinessNext \(bnext.com.tw\)](#)
- [????PDF Chatbot with Llama3 & RAG?? #chatbot #chatgpt #llama3 #rag #chatpdf - YouTube](#)
- [????????https://github.com/Shubhamsaboo/awesome-llm-apps](#)
- [Easy AI/Chat For Your Docs with Langchain and OpenAI in Python](#)
- [RAG????16?????????RAG](#)
- [YT:RAG????16?????????RAG - YouTube](#)
- [?? LLM ????-Day26-? Langchain ?? PDF ???? - iT ??::???????????? IT ???? \(ithome.com.tw\)](#)
- [RAG????LangChain + Llama2 |?????LLM | by ChiChieh Huang | Medium](#)
- [Python RAG Tutorial \(with Local LLMs\): AI For Your PDFs - YouTube](#)
- [? PDF ??????????????????](#)

Embedding/Rerank Models

- [????????](#)
- ??
 - [BCEmbedding](#)
 - HuggingFace: https://huggingface.co/maidalun1020/bce-embedding-base_v1
 - [BAAI](#)
 - [GTE](#)
- API Service
 - [Cohere](#) (Rerank)

Vector Databases

- [Qdrant - ??????????????????????????????GUI?????](#)
 - [???Qdrant????????????????? – AI StartUps Product Information, Reviews, Latest Updates](#)
- [Chroma](#)
 - Doc: [? Getting Started | Chroma Docs](#)
 - [Chroma?????????. ?????????????????????????? | by Lemooljiang | Medium](#)
 - [Chroma with Docker](#)
- [VectorAdmin](#) - ?????????? (????????? OpenAI)

- GitHub: <https://github.com/Mintplex-Labs/vector-admin>
- YT: [VectorAdmin | The universal GUI for vector databases - YouTube](#)
- [VectorAdmin in Docker](#)
- [Pinecone](#) (Cloud)
 - [Introducing Pinecone Inference to streamline your AI workflow | Pinecone](#)
 - [multilingual-e5-large - Pinecone Docs](#)
- [Supabase](#) (Cloud)
- [Astra DB](#) (Cloud)
 - Doc: [Quickstart | Astra DB Serverless | DataStax Docs](#)

Advanced RAG

- [RAG ??? | 7 ????????? | ??? LLM. ?? LLM + RAG ?????????????????? RAG ??... | by ChiChieh Huang | Medium](#)
- ReRank
 - [RAG ??????ReRank????????????? | DataAgent](#)
- [Advanced RAG: MultiQuery and ParentDocument | RAGStack | DataStax Docs](#)
- [Advanced Retrieval With LangChain \(ipynb\)](#)
- [Advanced RAG Implementation using Hybrid Search, Reranking with Zephyr Alpha LLM | by Nadika Poudel | Medium](#)
- [Five Levels of Chunking Strategies in RAG | Notes from Greg's Video | by Anurag Mishra | Medium](#)
- Chunking/Splitting
 - [Mastering RAG: Advanced Chunking Techniques for LLM Applications - Galileo \(rungalileo.io\)](#)
 - [5 Levels Of Text Splitting \(ipynb\)](#)
 - [??] [Semantic Chunking](#)
 - [????????? RAG ? Chunking ?????](#)
 - [Chunking Evaluation](#)
 - Online Tools
 - [Online Text Splitter](#)
 - [ChunkViz](#)
- [Advanced RAG: Query Expansion](#)

RAG Projects

- [Dot](#)

- [ragapp](#)
- [RAGFlow](#)
 - YT: [RAGFlow???????? - YouTube](#)
- [R2R](#)
- [Easy-RAG](#)
- [Langchain-Chatchat](#)
- [kotaemon](#)

Danswer

[Danswer](#) is the AI Assistant connected to your company's docs, apps, and people. Danswer provides a Chat interface and plugs into any LLM of your choice. Danswer can be deployed anywhere and for any scale - on a laptop, on-premise, or to cloud.

- GitHub: <https://github.com/danswer-ai/danswer>
- Doc: <https://docs.danswer.dev/introduction>

Embedchain

Embedchain streamlines the creation of personalized LLM applications, offering a seamless process for managing various types of unstructured data.

- Doc: [? Quickstart - Embedchain](#)
- GitHub: <https://github.com/embedchain/embedchain>

GraphRAG

??GraphRAG
 ???

- [Get Started](#)
- GitHub: <https://github.com/microsoft/graphrag>
- YT: [Microsoft GraphRAG | ????????RAG???????????????? - YouTube](#)
- GitHub: [GraphRAG Local with Ollama and Gradio UI](#)
- YT: [????RAG?GraphRAG?????????Gemma 2+Nomic Embed?????????GraphRAG+Chainlit+Ollama??? #graphrag #ollama #ai - YouTube](#)
- GitHub: [GraphRAG + AutoGen + Ollama + Chainlit UI = Local Multi-Agent RAG Superbot](#)

[neo4j](#)

- Doc: [GenAI Ecosystem - Neo4j Labs](#)
- ???: [??? AI ??????GraphRAG ????????????? LLM ????? | T?? \(techbang.com\)](#)
- [NeoConverse - Graph Database Search with Natural Language - Neo4j Labs](#)
- LangChain: [Enhancing RAG-based application accuracy by constructing and leveraging knowledge graphs \(langchain.dev\)](#)
- [Build a Question Answering application over a Graph Database | ??? LangChain](#)
- LangChain: <https://neo4j.com/labs/genai-ecosystem/langchain/>
- <https://github.com/neo4j-labs/llm-graph-builder>
- ipynb:
https://github.com/tomasonjo/blogs/blob/master/llm/enhancing_rag_with_graph.ipynb

Verba

Verba is a fully-customizable personal assistant for querying and interacting with your data, either locally or deployed via cloud. Resolve questions around your documents, cross-reference multiple data points or gain insights from existing knowledge bases. Verba combines state-of-the-art RAG techniques with Weaviate's context-aware database. Choose between different RAG frameworks, data types, chunking & retrieving techniques, and LLM providers based on your individual use-case.

- Github: [Retrieval Augmented Generation \(RAG\) chatbot powered by Weaviate](#)
- [Weaviate](#) is an open source, AI-native vector database
 - Doc: [Quickstart Tutorial | Weaviate - Vector Database](#)
- Video: [Open Source RAG with Ollama - YouTube](#)

PrivateGPT

- [Introduction – PrivateGPT | Docs](#)
- GitHub: <https://github.com/zylon-ai/private-gpt>
- Video: [PrivateGPT 2.0 - FULLY LOCAL Chat With Docs \(PDF, TXT, HTML, PPTX, DOCX, and more\) - YouTube](#)
- Video: [Installing Private GPT to interact with your own documents!! - YouTube](#)

LLMWare

The Ultimate Toolkit for Enterprise RAG Pipelines with Small, Specialized Models.

- YT: [??Mistral V3?????????? ?????? #ai #llm #mistral #mistral7b #finetune #???? #???? #nlp #embedding - YouTube](#)
- Colab: [How to Finetune Llama-3 and Export to Ollama | Unsloth Docs](#)
- Colab: [Fine-Tuning Llama-2 LLM on Google Colab: A Step-by-Step Guide. | by Gathnex | Medium](#)

Atlas

[Atlas by NOMIC](#) - ??????????????????????

AnythingLLM

?? Chat/Fine-Tune/Multi-Model ???????

- [??????LLMs????????? 20231228 - YouTube](#)

LLaMA-Factory

- GitHub: <https://github.com/hiyouga/LLaMA-Factory>
- YT: [Llama3 ?????????????????????? - YouTube](#)
- YT: [?LLaMA-Factory?????????????? ??????????????????WebUI ?????????????????????? - YouTube](#)

outlines

????????????????????????????????????

- GitHub: <https://github.com/outlines-dev/outlines>

Models

Gemini-Pro

??? Gemini-Pro ??????????????? Gemini API ?????????[Google AI Studio](#)?[Python SDK](#)?[REST API \(curl\)](#)?

- [??????? | Google AI for Developers | Google for Developers](#)

Mistral

?? [Mistral AI](#) ?????? SDK ? API?

- GitHub: [mistralai/mistral-finetune \(github.com\)](https://github.com/mistralai/mistral-finetune)
- [Mistral??????API?SDK????? | iThome](#)

AI Dev

AI Develop Frameworks

- [LangChain](#) (python + node.js)
- [LlamaIndex](#) (python)
- [Haystack](#) (python)

OpenUI

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

- GitHub: <https://github.com/wandb/openui>
- Video: [OpenUI & Llama 3: Effortless Text to Frontend UI Generation - YouTube](#)

Data Analysis (Chat with CSV)

- LangChain: [Chat with a CSV | LangChain Agents Tutorial \(Beginners\) - YouTube](#)
- PandasAI: [Multi-ChatCSV Streamlit App: Analyze Multiple CSV files with PandasAI and OpenAI | Step by Step - YouTube](#)
- Streamlit: [Chat with CSV Streamlit Chatbot using Llama 2: All Open Source - YouTube](#)
- [DataLine](#)
 - GitHub: <https://github.com/RamiAwar/dataline>

Chat with Dataset

- [Chat with your dataset! This project demonstrates a web-based application to query a dataset through natural language.](#)

PandasAI

[PandasAI](#) is a Python library that makes it easy to ask questions to your data in natural language. It helps you to explore, clean, and analyze your data using generative AI.

- GitHub: <https://github.com/Sinaptik-AI/pandas-ai>

- Video: [Multi-ChatCSV Streamlit App:Analyze Multiple CSV files with PandasAI and OpenAI| Step by Step - YouTube](#)

Web Scraper

- Crawlee

A web scraping and browser automation library.

- [crawlee for Python](#)
 - GitHub: <https://github.com/apify/crawlee-python>
- [crawlee for Node.js](#)
 - GitHub: <https://github.com/apify/crawlee>

- ScrapeGraphAI

ScrapeGraphAI is a open-source web scraping python library designed to usher in a new era of scraping tools.

- [Indices and tables — ScrapeGraphAI documentation](#)
- GitHub: <https://github.com/VinciGit00/Scrapegraph-ai>
- Video: [Scrape Any Website using llama3+Ollama+ScrapeGraphAI | Fully Local + Free #ai #llm - YouTube](#)

- Crew AI

[Crew AI](#) is a collaborative working system designed to enable various artificial intelligence agents to work together as a team, efficiently accomplishing complex tasks. Each agent has a specific role, resembling a team composed of researchers, writers, and planners.

- GitHub: <https://github.com/joaomdmoura/crewAI>
- [???? AI ??????????](#)
 - Video: [??????Agent??](#)
 - GitHub: [Python ???](#)
- [Crew AI — your own minions. How I Made AI Assistants Do My Work For... | by Csakash | Medium](#)

OpenAI API

- [What is ChatGPT API? - GeeksforGeeks](#)

- [\[D7\] OpenAI API ?? - ?????? - iT ????:???????????? IT ???? \(ithome.com.tw\)](#)
- [\[D8\] OpenAI API ?? - Chat Completion ???? - iT ????:???????????? IT ???? \(ithome.com.tw\)](#)
- [Model Parameters in OpenAI API. Let's break down the... | by Csakash | Medium](#)

Web UI Framework

- Gradio

Gradio is the fastest way to demo your machine learning model with a friendly web interface so that anyone can use it, anywhere!

- [Gradio](#)
- [Create a ChatBot with OpenAI and Gradio in Python - GeeksforGeeks](#)
- [????? OpenAI ChatBot ????](#)

- Streamlit

Streamlit is the UI powering the LLM movement

- [Streamlit • A faster way to build and share data apps](#)
- [AI talks: ChatGPT assistant via Streamlit](#)
- GitHub: [Some Example Codes](#)

AI Memory

- [How To Give Your Chatbot More Memory | by Dan Cleary | Medium](#)
- [\[D8\] OpenAI API ?? - Chat Completion ???? - iT ????:???????????? IT ???? \(ithome.com.tw\)](#)

VS Code

- [CodeGPT](#) - Code like a pro with our AI Copilot!
 - Doc: <https://docs.codegpt.co/docs/intro>
 - VSCode: [CodeGPT: Chat & AI Agents - Visual Studio Marketplace](#)
- [Continue](#)
 - [An entirely open-source AI code assistant inside your editor \(continue.dev\)](#)
 - Doc: <https://docs.continue.dev/intro>
 - VSCode: [Continue - Codestral, GPT-4o, and more - Visual Studio Marketplace](#)

- GitHub: <https://github.com/continuedev/continue>
- [Codeium](#) - ?? vim/Neovim ???????????? OpenAI API?????????
- VSCode: [Codeium: AI Coding Autocomplete and Chat for Python, Javascript, Typescript, Java, Go, and more - Visual Studio Marketplace](#)
- GitHub: <https://github.com/Exafunction/codeium.vim>
- Video: [?Arduino ?????AI ????? Codeium ????? Arduino? - YouTube](#)

Instructor

Structured outputs powered by llms. Designed for simplicity, transparency, and control.

- [Welcome To Instructor - Instructor \(jxnl.github.io\)](#)

Phidata

Phidata adds memory, knowledge and tools to LLMs.

- [What is phidata? - Phidata](#)

PDF Extractor

- [gptpdf](#) - ?? OpenAI API ?? PDF ??????? Markdown ???
- [omniparse](#) - PDF to Markdown
 - GitHub: <https://github.com/adithya-s-k/omniparse>
- [PDF-Extract-Kit](#) - Layout Detection, Formula Detection, Formula Recognition
- [Marker](#) - Marker converts PDF to markdown quickly and accurately.
 - Video: [Marker: This Open-Source Tool will make your PDFs LLM Ready - YouTube](#)
- [Mathpix](#) (cloud)

Responsible AI

- [Input Output Guardrails with Llama](#) (ipynb)

Gemini API

- Doc: [Gemini API ???????? | Google AI for Developers | Google for Developers](#)
- [Gemini API Cookbook](#)

LlamaIndex

- [Ollama <> Mistral <> LlamaIndex Cookbook](#) (ipynb)

- [???????????? - ??? - ALPHA Camp](#)
- [????? AI ?????????????? YouTube ?? - ALPHA Camp](#)
- [AI?????????AI???????????? - ALPHA Camp](#)

Medium Articles

- [ChiChieh Huang – Medium](#)

Course/HandBook

Google AI Courses for Free

- [Beginner: Introduction to Generative AI Learning Path](#)
- [Machine Learning | Resources | Google for Developers](#)

Microsoft

- [Microsoft Learn](#)
- [Generative AI for Beginners \(microsoft.github.io\)](#)

????(NCHC)??

- [????????????Taichung.py 2024/04/23 Meetup, ?????????????????????????????? - YouTube](#)
- [??????LLMs???????? 20231228 - YouTube](#)
- [?????-\[??\]????RAG???????????????? - YouTube](#)
- [?????-\[??\]????RAG???????????????? Q&A - YouTube](#)

LLM Tokenizer ???

- [OpenAI Platform](#)
- [The Tokenizer Playground - a Hugging Face Space by Xenova](#)

PyImageSearch ?? (??)

- 1: [Harnessing Power at the Edge: An Introduction to Local Large Language Models - PyImageSearch](#)
- 2: [Inside Look: Exploring Ollama for On-Device AI - PyImageSearch](#)

- 3: [Integrating Local LLM Frameworks: A Deep Dive into LM Studio and AnythingLLM - PyImageSearch](#)
- 4: [Exploring Oobabooga Text Generation Web UI: Installation, Features, and Fine-Tuning Llama Model with LoRA - PyImageSearch](#)

AI ????????

- [Toolify.ai](#) - ?????????? AI ??????????????
- [?????????LLM?????](#)
- [llm-course](#) - ?????????? AI ??????????????????????
- [?? ?? LLM ???](#)

???

[?????????? | ??? \(iii.org.tw\)](#)

- ???AI???????????
- ???AI???????????
- 2023????????AI??-????AI?????

RedHat AI

Red Hat® Enterprise Linux® AI is a foundation model platform to seamlessly develop, test, and run Granite family large language models (LLMs) for enterprise applications.

Red Hat Enterprise Linux AI brings together:

- The Granite family of open source-licensed LLMs, distributed under the Apache-2.0 license with complete transparency on training datasets.
- InstructLab model alignment tools, which open the world of community-developed LLMs to a wide range of users.
- A bootable image of Red Hat Enterprise Linux, including popular AI libraries such as PyTorch and hardware optimized inference for NVIDIA, Intel, and AMD.
- Enterprise-grade technical support and model intellectual property indemnification provided by Red Hat.

URLs:

- [Red Hat Enterprise Linux AI](#)
- [Red Hat Delivers Accessible, Open Source Generative AI Innovation with Red Hat Enterprise Linux AI](#)

InstructLab

Command-line interface. Use this to chat with the model or train the model (training consumes the taxonomy data)

What are the components of the InstructLab project?

- **Taxonomy**
InstructLab is driven by taxonomies, which are largely created manually and with care. InstructLab contains a taxonomy tree that lets users create models tuned with human-provided data, which is then enhanced with synthetic data generation.
- **Command-line interface (CLI)**
The InstructLab CLI lets contributors test their contributions using their laptop or workstation. Community members can use the InstructLab technique to generate a low-fidelity approximation of synthetic data generation and model-instruction tuning without access to specialized hardware.
- **Model training infrastructure**
Finally, there's the process of creating the enhanced LLMs. It takes GPU-intensive infrastructure to regularly retrain models based on new contributions from the community. IBM donates and maintains the infrastructure necessary to frequently

retrain the InstructLab project's enhanced models.

How is InstructLab different from retrieval-augmented generation (RAG)?

RAG is a cost-efficient method for supplementing an LLM with domain-specific knowledge that wasn't part of its pretraining. RAG makes it possible for a chatbot to accurately answer questions related to a specific field or business without retraining the model. Knowledge documents are stored in a vector database, then retrieved in chunks and sent to the model as part of user queries. This is helpful for anyone who wants to add proprietary data to an LLM without giving up control of their information, or who needs an LLM to access timely information.

This is in contrast to the InstructLab method, which sources end-user contributions to support regular builds of an enhanced version of an LLM. InstructLab helps add knowledge and unlock new skills of an LLM.

It's possible to "supercharge" a RAG process by using the RAG technique on an InstructLab-tuned model.

URLs:

- [What is InstructLab? \(redhat.com\)](#)
- [InstructLab Community](#)
- [Quick Start Guide](#)
- GitHub: [taxonomy](#)
- Docs: [taxonomy](#)
- [InstructLab Community Collaboration Spaces](#)

Agent

Camel AI

CAMEL-AI.org is the 1st LLM multi-agent framework and an open-source community dedicated to finding the scaling law of agents.

- GitHub: <https://github.com/camel-ai/camel>

Crew AI

[Crew AI](#) is a collaborative working system designed to enable various artificial intelligence agents to work together as a team, efficiently accomplishing complex tasks. Each agent has a specific role, resembling a team composed of researchers, writers, and planners.

- GitHub: <https://github.com/joaomdmoura/crewAI>
- [???? AI ??????????](#)
 - Video: [??????Agent??](#)
 - GitHub: [Python ???](#)
- [Crew AI — your own minions. How I Made AI Assistants Do My Work For... | by Csakash | Medium](#)

SWE-agent

SWE-agent turns LMs (e.g. GPT-4) into software engineering agents that can fix bugs and issues in real GitHub repositories.

- [SWE-agent documentation \(princeton-nlp.github.io\)](#)

AutoGen

Enable Next-Gen Large Language Model Applications

- [AutoGen | AutoGen \(microsoft.github.io\)](#)
- GitHub: <https://github.com/microsoft/autogen>

AutoGPT

AutoGPT is the vision of accessible AI for everyone, to use and to build on. Our mission is to provide the tools, so that you can focus on what matters.

- [The Official Auto-GPT Website \(agpt.co\)](https://agpt.co)
- GitHub: <https://github.com/Significant-Gravitas/AutoGPT>

AutoGPT-Code-Ability

AutoGPT's coding ability is an open-source coding assistant powered by AI. The goal is to make software development more accessible to everyone, regardless of skill level or resources. By generating code in Python, a popular and very accessible language, AutoGPT acts as a virtual co-pilot to help users build projects like backends for existing frontends or command-line tools.



- GitHub: <https://github.com/Significant-Gravitas/AutoGPT-Code-Ability>

- [AgentOps AI](#)
Replay Analytics and Debugging, LLM Cost Management, Agent Benchmarking
- [Langtrace](#)


```
fib(n) = fib(n-1) + fib(n-2)
```

Code Review

Prompt

Where is the bug in this code?

```
def fib(n):  
    if n <= 0:  
        return n  
    else:  
        return fib(n-1) + fib(n-2)
```

Response:

The bug in this code is that it does not handle the case where `n` is equal to 1.

Unit Tests

Prompt

write a unit test for this function: \$(cat fib.py)

Response:

Here's an example of how you could write a unit test for the `fib` function:

```
...  
  
import unittest  
  
class TestFib(unittest.TestCase):  
    def test_fib(self):  
        self.assertEqual(fib(0), 0)  
        self.assertEqual(fib(1), 1)  
        self.assertEqual(fib(2), 1)  
        self.assertEqual(fib(3), 2)  
        self.assertEqual(fib(4), 3)  
        self.assertEqual(fib(5), 5)  
        self.assertEqual(fib(6), 8)  
        self.assertEqual(fib(7), 13)  
        self.assertEqual(fib(8), 21)
```

```
self.assertEqual(fib(9), 34)
if __name__ == '__main__':
    unittest.main()
'''
```

This test case covers all the cases where the input is an integer between 0 and 9, inclusive.

Code Completion

```
# Prompt
# A simple python function to remove whitespace from a string:
```

Response:

```
def remove_whitespace(s):
    return ".join(s.split())
```

Infill: `<PRE> {prefix} <SUF> {suffix} <MID>`

```
# Prompt
<PRE> def compute_gcd(x, y): <SUF>return result <MID>
```

Response:

```
if x == y:
    return x

if x > y:
    x = x - y
else:
    y = y - x

result = compute_gcd(x, y)
```

[CURSOR](#) Prompt

- [Cursor Directory](#)

```
CURSOR_CHAT_PROMPT = ""
System: You are an intelligent programmer, powered by GPT-4. You are happy to help answer any questions that
```

the user has (usually they will be about coding).

1. Please keep your response as concise as possible, and avoid being too verbose.
2. When the user is asking for edits to their code, please output a simplified version of the code block that highlights the changes necessary and adds comments to indicate where unchanged code has been skipped. For example:

```
```file_path
// ... existing code ...
{{ edit_1 }}
// ... existing code ...
{{ edit_2 }}
// ... existing code ...
...`
```

The user can see the entire file, so they prefer to only read the updates to the code. Often this will mean that the start/end of the file will be skipped, but that's okay! Rewrite the entire file only if specifically requested. Always provide a brief explanation of the updates, unless the user specifically requests only the code.

3. Do not lie or make up facts.
4. If a user messages you in a foreign language, please respond in that language.
5. Format your response in markdown.
6. When writing out new code blocks, please specify the language ID after the initial backticks, like so:

```
```python
{{ code }}
...`
```

7. When writing out code blocks for an existing file, please also specify the file path after the initial backticks and restate the method / class your codeblock belongs to, like so:

```
```typescript:app/components/Ref.tsx
function AIChatHistory() {{
 ...
 {{ code }}
 ...
}}
...`
```

User: Please also follow these instructions in all of your responses if relevant to my query. No need to acknowledge these instructions directly in your response.

<custom\_instructions>

Respond the code block in English!!!! this is important.

</custom\_instructions>

## Current File

Here is the file I'm looking at. It might be truncated from above and below and, if so, is centered around my cursor.

```\${file\_path}

{file_contents}

```

{user\_message}

""

CURSOR\_REWRITE\_PROMPT = ""

System: You are an intelligent programmer. You are helping a colleague rewrite a piece of code.

Your colleague is going to give you a file and a selection to edit, along with a set of instructions. Please rewrite the selected code according to their instructions.

Think carefully and critically about the rewrite that best follows their instructions.

The user has requested that the following rules always be followed. Note that only some of them may be relevant to this request:

## Custom Rules

Respond the code block in English!!!! this is important.

User: First, I will give you some potentially helpful context about my code.

Then, I will show you the selection and give you the instruction. The selection will be in `\${file\_path}`.

-----

## Potentially helpful context

#### file\_context\_4

{file\_context\_4}

```
file_context_3
```

```
{file_context_3}
```

```
file_context_2
```

```
{file_context_2}
```

```
file_context_1
```

```
{file_context_1}
```

```
file_context_0
```

```
{file_context_0}
```

This is my current file. The selection will be denoted by comments "Start of Selection" and "End of Selection":

```
```${file_path}
```

```
# Start of Selection
```

```
{code_to_rewrite}
```

```
# End of Selection
```

Please rewrite the selected code according to the instructions.

Remember to only rewrite the code in the selection.

Please format your output as:

```
```
```

```
Start of Selection
```

```
INSERT_YOUR_REWRITE_HERE
```

```
End of Selection
```

Immediately start your response with

```
```
```

```
'''
```

For RAG

Rewrite User's Question

Given the following conversation, rewrite the last user input to reflect what the user is actually asking.

be used to provide a roadmap for future crimes.

- Help people understand how systems work that are used to prevent crimes, provided those descriptions do not provide detailed information on how to defeat those systems.

O4: Guns and Illegal Weapons.

Should not

- Assist in the planning or execution of firearm or illegal weapons crimes, especially when the person indicates that they may act on the information provided to them.
- Encourage people to engage in firearm or illegal weapons crimes or condone such behavior.

Can

- Discuss firearms and the arguments for and against firearm ownership.

O5: Self-Harm.

Should not

- Encourage people to harm themselves.
- Romanticize or condone self-harm or suicide.
- Provide information regarding the methods of suicide or self-harm.
- Provide any information or advice, other than information regarding appropriate health resources, to people who express or imply that they wish to harm themselves.

Can

- Provide general information regarding self-harm and suicide (ex: statistics, the mental health challenges faced by those who take these actions).

Should

- Provide information regarding appropriate health resources to people who imply or express the desire to harm themselves.

<END UNSAFE CONTENT CATEGORIES>

<BEGIN CONVERSATION>

\$prompt

<END CONVERSATION>

Provide your safety assessment for \$agent_type in the above conversation:

- First line must read 'safe' or 'unsafe'.
- If unsafe, a second line must include a comma-separated list of violated categories.

Cheat Sheet

The ChatGPT Cheat Sheet V2

Basic Structure of a Prompt:

Acting as a [ROLE] perform [TASK] in [FORMAT]

Act as a [ROLE]

Marketer
Advertiser
Copywriter
Accountant
Lawyer
Analyst
Ghostwriter
Project Manager
Therapist
Journalist
Chief Financial Officer
Prompt Engineer

Create a [TASK]

Headline
Presentation / Webinar
Essay
Book Outline
Email Sequence
Social Media Post
Product Description
Cover Letter / Resume
Blog Post / Article
Summary
Tiktok, YT Reel - Video Script
Sales Page / Ad Copy

show as [FORMAT]

A Table
A list
Summary
HTML
Code
Spreadsheet
CSV file
Plain Text file
Rich Text
PDF
Markdown
Word Cloud

Prompt Chaining

- 1 - Provide me with the ideal outline for an effective & persuasive blog post.
- 2 - Write a list of Engaging Headlines for this Blog post based on [Topic].
- 3 - Write a list of Subheadings & Hooks for this same blog post
- 4 - Write a list of Keywords for this Blog.
- 5 - Write a list of Compelling Call-to-Actions for the blog post
- 6 - Combine the best headline with the best Subheadings, Hooks, Keywords and Call-to-Action to write a blog post for [topic]
- 7 - Re-write this Blog Post in the [Style], [Tone], [Voice] and [Personality].

Tree of Thought Prompt

Three experts with exceptional logical thinking skills are collaboratively answering a question using a tree of thoughts method.

Each expert will share their thought process in detail, taking into account the previous thoughts of others and admitting any errors. They will iteratively refine and expand upon each other's ideas, giving credit where it's due.

The process continues until a conclusive answer is found. Use step by step thinking & organize the entire response in detailed steps in a markdown table format. Once this table is complete, provide a summary of the proposed recommendations. My question is...

Mind-Blowing Meta Prompt

What are the absolute coolest, most mind-blowing, out of the box, ChatGPT prompts that will really show off the power of ChatGPT? Give me 10.

The prompts should focus on combining [topic1] & [topic2] for [main outcome].

Mimic Your Writing Style

Acting as an Expert Ghostwriter assess the tone, style, voice, personality, Perplexity & Burstiness used in the following sample content so you can mimic this.

Perplexity measures the complexity of text. Separately, burstiness compares the variations of sentences. Humans tend to write with greater burstiness, for example, with some longer or complex sentences alongside shorter ones.

Provide a table outlining the description for Tone, Style, Voice, Personality, Perplexity & Burstiness based on the sample content.

When the table is finished, I will need the assessment combined into a statement designed in a way that it could be used in a prompt to get ChatGPT to simulate my own writing.

Over-The-Top Creative Copy

Take this statement and make it explode with hyperbole and copy-writing cinematography.

The statement is:
[insert statement]

These prompts are designed to be used with ChatGPT Pro (GPT4). Using them on ChatGPT 3.5 (free version) might not generate the highest quality outcomes.

Made: June 2023 @shaneozard

Function Calling

LLM?Large Language Model???????? Function
Calling????????????????LLM??
?????
Function Calling ?????????LLM ?????????? reconocize
??LLM?LLM
??

??
?????????LLM????????????????LLM ??????????????????????API?????????????
LL??

??
Function Calling ??????????????????????
• ??????LLM ???
• API ???LLM ?????????API?????API?????API????????????????
• ??????LLM ???
• ??????LLM ??business logic ??

???Function Calling ? LLM ?? Complex ???????????

Tutorials

- [9 Best Local LLM For Function Calling \(Open Source\) \[2024\] - Sci Fi Logic](#)

Models

- [Berkeley Function Calling Leaderboard \(aka Berkeley Tool Calling Leaderboard\)](#)

Python Coding

LLM Model API

LMStudio

```
from langchain.llms import OpenAI

#set llm for langchain using model from lmstudio
llm = OpenAI(
    openai_api_base='http://localhost:1234/v1',
    openai_api_key='NULL'
)
```

```
import streamlit as st
from openai import OpenAI

# Set up the Streamlit App
st.title("ChatGPT Clone using Llama-3 🦙")
st.caption("Chat with locally hosted Llama-3 using the LM Studio 🦙")

# Point to the local server setup using LM Studio
client = OpenAI(base_url="http://localhost:1234/v1", api_key="lm-studio")

# Initialize the chat history
if "messages" not in st.session_state:
    st.session_state.messages = []

# Display the chat history
for message in st.session_state.messages:
    with st.chat_message(message["role"]):
        st.markdown(message["content"])

# Accept user input
if prompt := st.chat_input("What is up?"):
    # Add user message to chat history
```

```

st.session_state.messages.append({"role": "user", "content": prompt})
# Display user message in chat message container
with st.chat_message("user"):
    st.markdown(prompt)
# Generate response
response = client.chat.completions.create(
    model="lmstudio-community/Meta-Llama-3-8B-Instruct-GGUF",
    messages=st.session_state.messages, temperature=0.7
)
# Add assistant response to chat history
st.session_state.messages.append({"role": "assistant", "content": response.choices[0].message.content})
# Display assistant response in chat message container
with st.chat_message("assistant"):
    st.markdown(response.choices[0].message.content)

```

GPT

```

from langchain_openai import ChatOpenAI

llm = ChatOpenAI(
    model="gpt-4o",
    temperature=0,
    max_tokens=None,
    timeout=None,
    max_retries=2,
    # api_key="...",
    # base_url="...",
    # organization="...",
    # other params...
)

```

Ollama

```

from langchain_community.llms import Ollama

llm = Ollama(model="llama2:13b")
llm.invoke("The first man on the moon was ... think step by step")

```

Chunking/Splitting

??????

```
# Unicode []
# \u3002 []
# \uff0c []
# Get Unicode for specific character
# >>> ' '.encode('unicode-escape') # for py3
# >>> list(u' ') # for py2

import re
text = "[]"
chunks = re.split('[\u3002\u002c]', text)
#print("\n\n".join([chunk for chunk in chunks]))
for chunk in chunks:
    print("----" * 10)
    print(chunk)
```

??????

```
# \s+ []
chunks = re.split(r'(?<=[?!])\s+', text)
```

LLM Engine

A software that can load the LLM Models

LLM Engine

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
```


LLM Engine

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/>
- <https://github.com/Mintplex-Labs/anything-llm>

LLM Engine

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](https://www.youtube.com/watch?v=881111111111)

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](#)

LLM Engine

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>

Text generation web UI

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) ??????

LLM Engine

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='')
```

LLM Engine

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.

URLs

- [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](#)

LLM Engine

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>

Bechmark

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?"
]
```

```

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):

```

```

"""
████████████████████

class:
    load_url (str): ████████URL█
    max_concurrent_requests (int): ████████
    total_requests (int): ████████

class:
    tuple: (int) token ██████████
"""

# Semaphore ██████████
sem = asyncio.Semaphore(max_concurrent_requests)

# aiohttp 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)

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

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

    # (int) token ██████████
    return completion_tokens, response_times

if __name__ == '__main__':
    import sys

```

```

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}')

```


Translator

?? LLM ???????

LiteLLM + ????? + ???

- GitHub: https://github.com/wshuyi/workflows_with_litellm_pub

Translation Agent

- GitHub: <https://github.com/andrewyng/translation-agent>

RTranslator

RTranslator is an ([almost](#)) open-source, free, and offline real-time translation app for Android.

- GitHub: <https://github.com/niedev/RTranslator>

Jupyter Notebook

Installation

With pip

```
pip install notebook
```

Python Virtual Environment

With Python Venv

```
mkdir my-rag
cd my-rag
python -m venv .venv
source .venv/bin/activate
(my-rag)> pip install --upgrade pip
(my-rag)> pip install notebook
(my-rag)> jupyter notebook
```

With Conda

```
conda create -n my-rag python=3.10
conda activate my-rag
(my-rag)> pip install --upgrade pip
(my-rag)> pip install notebook
(my-rag)> jupyter notebook
```

UI ?????????????????????? ipykernel?

```
mkdir my-rag
cd my-rag
python -m venv .venv
source .venv/bin/activate
(my-rag)> pip install --upgrade pip
(my-rag)> pip install ipykernel
(my-rag)> ipython kernel install --user --name="my-rag-kernel"
```

(my-rag)> jupyter notebook

Cloud Providers

- [Google Colab](#)


```

from langchain_core.runnables import RunnablePassthrough

template = """Answer the question based only on the following context:

{context}

Question: {question}
"""

prompt = ChatPromptTemplate.from_template(template)
model = ChatOpenAI()

def format_docs(docs):
    return "\n\n".join([d.page_content for d in docs])

chain = (
    {"context": retriever | format_docs, "question": RunnablePassthrough()}
    | prompt
    | model
    | StrOutputParser()
)

chain.invoke("What did the president say about technology?")

```

ChatPromptTemplate

```

few_shot_examples = [
    {"input": "Could you please clarify the terms outlined in section 3.2 of the contract?",
    "output": "Certainly, I will provide clarification on the terms in section 3.2."},
    {"input": "We are interested in extending the payment deadline to 30 days instead of the current 15 days. Additionally, we would like to add a clause regarding late payment penalties.",
    "output": "Our request is to extend the payment deadline to 30 days and include a clause on late payment penalties."},
    {"input": ""The current indemnification clause seems too broad. We would like to narrow it down to cover only direct damages and exclude consequential damages. Additionally, we propose including a dispute resolution clause specifying arbitration as the preferred method of resolving disputes.""",
    "output": ""We suggest revising the indemnification clause to limit it to covering direct damages and excluding

```

consequential damages.

Furthermore, we recommend adding a dispute resolution clause that specifies arbitration as the preferred method of resolving disputes.""}},

```
{"input": "I believe the proposed changes are acceptable.",
```

```
"output": "Thank you for your feedback. I will proceed with implementing the proposed changes."}
```

```
]
```

```
few_shot_template = ChatPromptTemplate.from_messages(
```

```
[
```

```
    ("human", "{input}"),
```

```
    ("ai", "{output}")
```

```
]
```

```
)
```

```
few_shot_prompt = FewShotChatMessagePromptTemplate(
```

```
    example_prompt=few_shot_template,
```

```
    examples=few_shot_examples,
```

```
)
```

```
print(few_shot_prompt.format())
```

Loader

Web

```
from langchain_community.document_loaders import WebBaseLoader
loader = WebBaseLoader(
    web_paths=("https://lilianweng.github.io/posts/2023-06-23-agent/"),
    bs_kwargs=dict(
        parse_only=bs4.SoupStrainer(
            class_=("post-content", "post-title", "post-header")
        )
    ),
)
docs = loader.load()
```

Text

```
from langchain_community.document_loaders import DirectoryLoader
loader = DirectoryLoader("../", glob="**/*.md")
```

```
docs = loader.load()
len(docs)
print(docs[0].page_content[:100])
```

```
from langchain.document_loaders import TextLoader

dataset_folder_path='/path/to/dataset/'
documents=[]
for file in os.listdir(dataset_folder_path):
    loader=TextLoader(dataset_folder_path+file)
    documents.extend(loader.load())

print(documents[:3])
```

Markdown

```
"""
%pip install "unstructured[md]"
"""

from langchain_community.document_loaders import UnstructuredMarkdownLoader
markdown_path = "../././README.md"
loader = UnstructuredMarkdownLoader(markdown_path)

data = loader.load()
assert len(data) == 1
readme_content = data[0].page_content
print(readme_content[:3])
```

PDF + Text

```
from langchain_community.document_loaders import TextLoader
from langchain_community.document_loaders import PyPDFLoader

documents = []
for filename in SAMPLEDATA:
    path = os.path.join(os.getcwd(), filename)

    if filename.endswith(".pdf"):
        loader = PyPDFLoader(path)
        new_docs = loader.load_and_split()
```

```
    print(f"Processed pdf file: {filename}")
elif filename.endswith(".txt"):
    loader = TextLoader(path)
    new_docs = loader.load_and_split()
    print(f"Processed txt file: {filename}")
else:
    print(f"Unsupported file type: {filename}")

if len(new_docs) > 0:
    documents.extend(new_docs)

SAMPLEDATA = []

print(f"\nProcessing done.")
```

?????

?????

```
# Helper function for printing docs
def pretty_print_docs(docs):
    print(
        f"\n{'-' * 100}\n".join(
            [f"Document {i+1}:\n\n" + d.page_content for i, d in enumerate(docs)]
        )
    )
```

Finance AI

OpenBB

Investment research made easy with AI.

- [Investment Research | OpenBB](#)
- GitHub: [openbb-agents](#)
- GitHub: [OpenBB Platform](#)

StockBot

- GitHub: <https://github.com/bklieger-groq/stockbot-on-groq>

FinGPT

- [AI4Finance-Foundation.org - FinGPT, FinRobot, FinRL, AI Agent, FinLLMs, Open-Source Libraries](#)
- <https://github.com/AI4Finance-Foundation/FinGPT>

Semantic Kernel

Semantic Kernel ?????????????? AI ?????(??)????????? AI ?????????? AI ?????????? C#?Python
? Java ???

??????

- [Introduction to Semantic Kernel | Microsoft Learn](#)
- [Video][??] [? .NET ??? - ?????????? Microsoft ???????????](#)
- GitHub: <https://github.com/microsoft/semantic-kernel>

??????

- [Semantic Kernel?????????????-??OpenAI?AOAI???](#)
- [Video][?????] <https://www.youtube.com/watch?v=sByJwdJhc3s&t=45s>