# 作者： 孙 升 （上海大学 材料基因组工程研究院）

# 相关背景

大模型从免费和开源的角度上，可以分为两类：1. 不开放使用的大模型，如GPT-4，Claude-3等；2. 开源大模型，可以下载到本地。

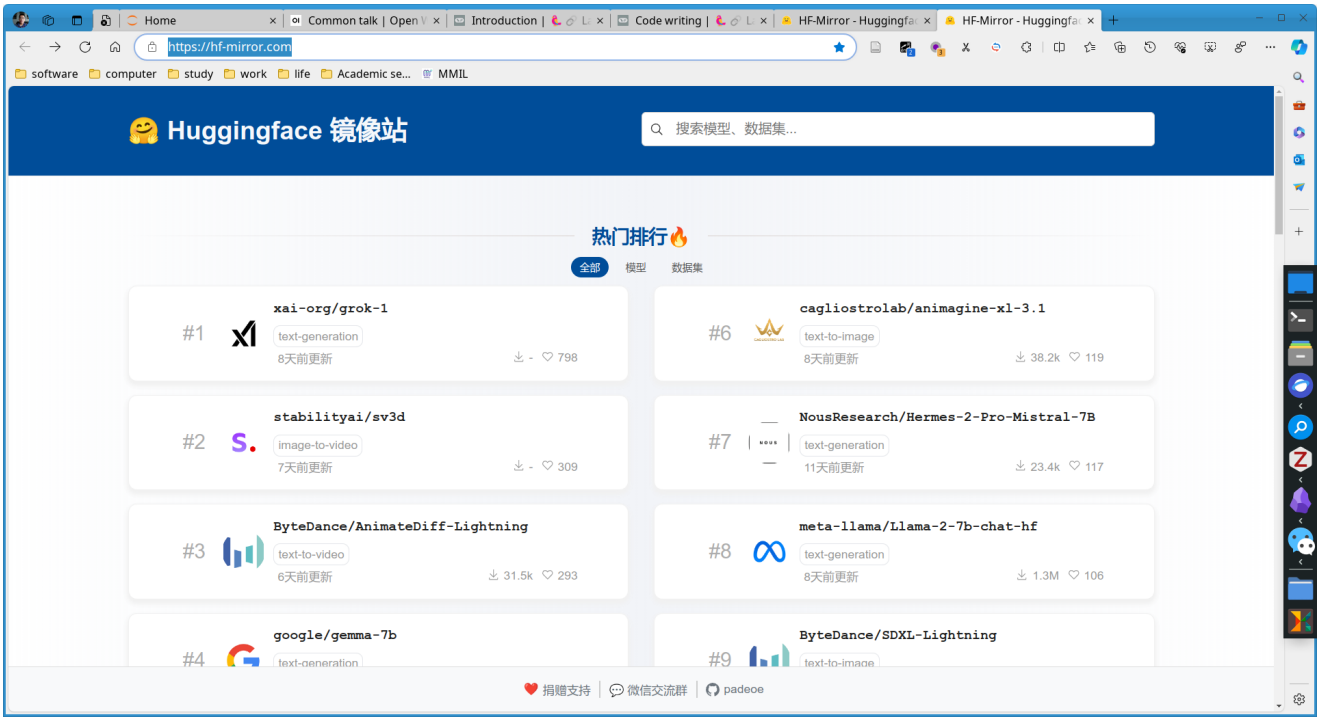
不开源的大模型，一般都是商业运行，需要用户交钱以后使用，用户缴纳购买的钱数后，会得到商业公司提供的API，一般还有商业公司自己页面上的聊天窗口。开源的大模型，可以下载到本地后，进行本地运行，可玩性更大，但是效果一般不如商业模型。

开源模型的效果不如商业模型的一个原因是，开源模型的参数量一般会远小于商业模型的。商业模型的参数量一般没有透漏，但是最近马斯克开源了他们的模型Grok-1，这个模型的参数量是314B（注：1B代表10亿参数量）。在这之前，发布的开源模型的最大参数量是70B。一般开源模型的参数量是2B，7B，13-16B，34B这四个量级。另外还有屈指可数的几个70B模型。

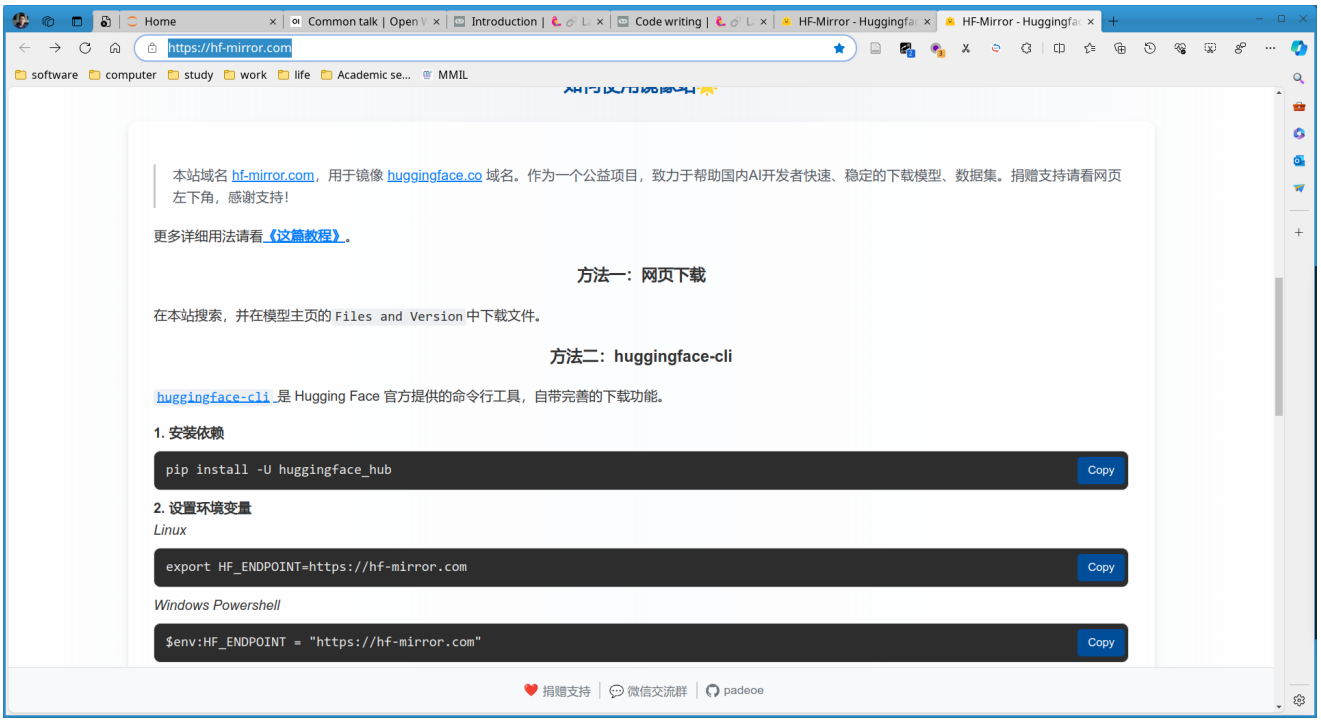
不过大多数研究人员的资源配置，也无法支持参数规模太大的模型。运行13B的原始模型，一般需要32G的及其内存或显存。

# 从哪儿下载大模型

现在有若干开源大模型的下载网站，笔者基本上是从huggingFace下载，使用国内的镜像网站：https://hf-mirror.com/。




镜像网站上列出了下载大模型的方法，此处不再赘述：




# 怎么启动运行大模型

## 在程序中调用相关的包，直接加载大模型。

例如可以调用python的transformers包，直接在程序中通过模型存储位置和模型名称加载并使用大模型。或者通过llama-cpp-python包加载和使用GGUF格式的量化模型。

除非是为了fine-tuning大模型等更改大模型参数的情形，一般这种使用方式较少。本介绍不涉及更改模型本身的参数，不讨论fine-tune大模型的工作（主要还是因为没有足够资源）。

## 通过某种程序加载模型，进行界面聊天方式（Chat方式）

这类的程序较多，笔者尝试使用过 GPT4All、lm-studio、text-generation-webui。这类软件可以在程序界面中选择要加载的本地大模型，并设置模型参数，进行问答式窗口的大模型使用。要说明的一点是，GPT4All和lm-studio是安装在本地的程序，它们尚未支持原生大小transformer格式的模型，着重支持GGUF量化格式的模型。text-generation-webui可以支持所有格式的模型，它是一个网络服务器，安装和运行后，在给定的端口通过网络浏览器打开界面，进行操作。text-generation-webui的功能更多，可以通过不同的对话方式开展对话: chat和文本方式等，还可以进行模型fine-tune。

## API方式

Server API的方式对两件事情特别重要：

1. 在程序编写中，不通过程序直接启动模型，就像使用GPT-4的API那样，直接调用已经在服务器端启动好的大模型，减少本地资源开销。一处布置，处处可用。
2. Serve端和client端分离。可以在客户端直接填入API地址，即可不占客户端资源的运行大模型。

本地模型也可以提供API接口。如上述的lm-studio和text-generation-webui，都是既可以通过chat方式聊天，也可以提给serve API。

另外一个是笔者最喜欢的运行方式：通过OLLAMA。从网站上下载了最新的对应于自己使用的操作系统的ollma程序后，可以通过

```
OLLAMA_HOST=0.0.0.0:your_port  /your_path_to_ollama/ollma serve
```

启动服务。your_port可以是机器上的任意未被占用端口，如`OLLMA_HOST=0.0.0.0:8887 /your_path_to_ollama/ollma serve`

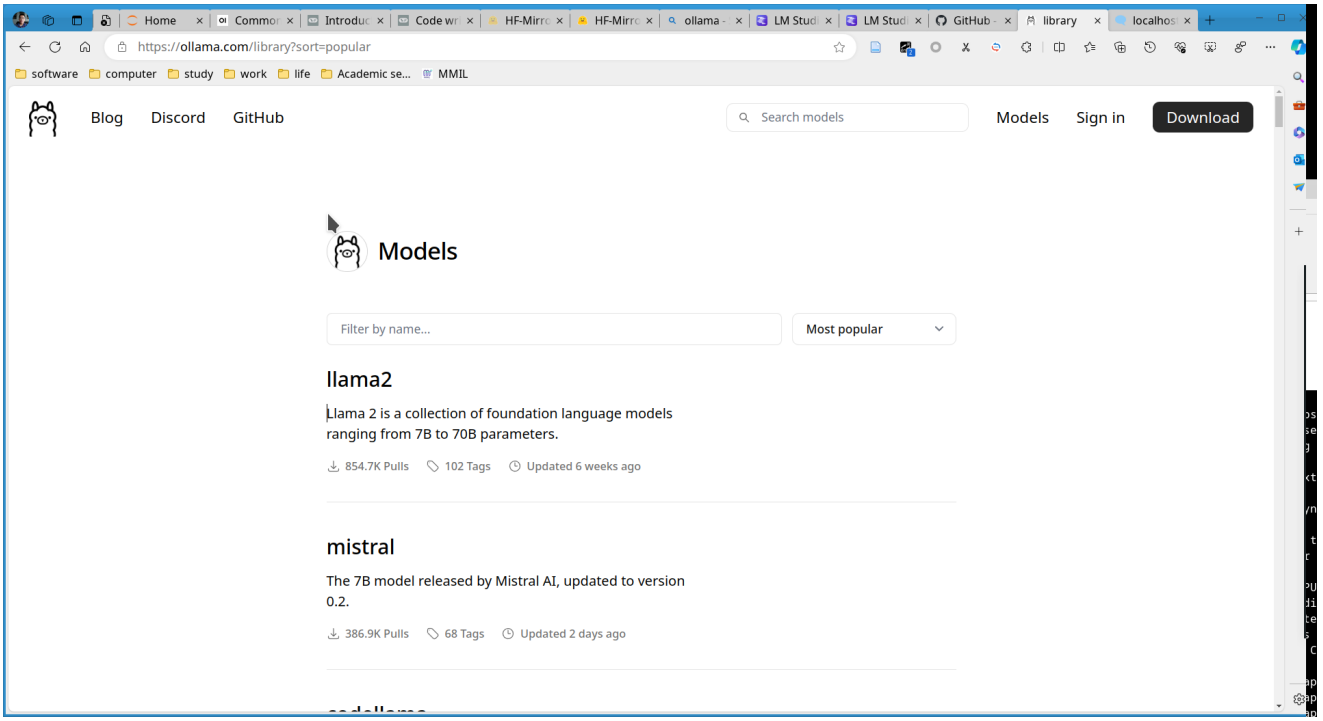
```
(base) ➜  LLM OLLAMA_HOST=0.0.0.0:8887 ~/LLM/ollama-linux-amd64 serve
time=2024-03-26T14:31:08.321+08:00 level=INFO source=images.go:710 msg="total blobs: 49"
time=2024-03-26T14:31:08.545+08:00 level=INFO source=images.go:717 msg="total unused blobs removed: 0"
time=2024-03-26T14:31:08.546+08:00 level=INFO source=routes.go:1021 msg="Listening on [::]:9997 (version 0
.1.28)"
time=2024-03-26T14:31:08.547+08:00 level=INFO source=payload_common.go:107 msg="Extracting dynamic librari
es..."
time=2024-03-26T14:31:16.703+08:00 level=INFO source=payload_common.go:146 msg="Dynamic LLM libraries [cpu
_avx2 cpu_avx rocm_v5 cuda_v11 cpu]"
time=2024-03-26T14:31:16.703+08:00 level=INFO source=gpu.go:94 msg="Detecting GPU type"
time=2024-03-26T14:31:16.703+08:00 level=INFO source=gpu.go:265 msg="Searching for GPU management library
libnvidia-ml.so"
time=2024-03-26T14:31:16.726+08:00 level=INFO source=gpu.go:311 msg="Discovered GPU libraries: [/usr/lib/l
ibnvidia-ml.so.535.104.05 /usr/lib32/libnvidia-ml.so.535.104.05 /usr/lib64/libnvidia-ml.so.535.104.05]"
time=2024-03-26T14:31:16.915+08:00 level=INFO source=gpu.go:99 msg="Nvidia GPU detected"
time=2024-03-26T14:31:16.915+08:00 level=INFO source=cpu_common.go:11 msg="CPU has AVX2"
time=2024-03-26T14:31:16.921+08:00 level=INFO source=gpu.go:146 msg="CUDA Compute Capability detected: 8.6
"
[GIN] 2024/03/26 - 14:34:19 | 200 |  299.281607ms |             ::1 | GET      "/api/tags"
[GIN] 2024/03/26 - 14:41:16 | 200 |    8.045217ms |             ::1 | GET      "/api/tags"
[GIN] 2024/03/26 - 14:41:16 | 200 |    3.585429ms |             ::1 | GET      "/api/tags"
[GIN] 2024/03/26 - 14:50:10 | 200 |     3.67745ms |             ::1 | GET      "/api/tags"
```

可以在浏览器中打开`localhost:8887`，此时会显示`Ollama is running`，代表服务启动成功。

服务启动后，就可以加载使用模型了。模型来源有两种，一种比较简单，直接执行

```
OLLMA_HOST=0.0.0.0:your_port /your_path_to_ollama/ollma run model_name
```

可以使用的model_name在这儿查找https://ollama.com/library




运行命令后，会先下载对应的模型，然后使用终端式的界面进行对话：

```
(base) ➜  ~ OLLAMA_HOST=0.0.0.0:9997 ~/LLM/ollama-linux-amd64 run nomic-embed-text:latest
Error: embedding models do not support chat
(base) ➜  ~ OLLAMA_HOST=0.0.0.0:9997 ~/LLM/ollama-linux-amd64 run yi:34b-chat-q6_K
>>>
Use Ctrl + d or /bye to exit.
>>> who are you
Hello! My name is Yi, and I am a language model based on the transformers architecture developed by
01.AI. My purpose is to be a helpful resource for you, capable of answering questions and offering
insightful information across a wide range of topics. How may I be of service to you today?

>>> Send a message (/? for help)
```

另一种模型的来源是从Huggingface下载的GGUF格式的量化模型，使用这种模型，需要先写一个modelfile，然后使用

```
OLLMA_HOST=0.0.0.0:your_port /your_path_to_ollama/ollma create  model_name -f
modelfile
```

来生成可以通过run命令运行的模型。

Modelfile有许多的例子，可以在ollama的网站讨论版上找到，也可以自己撰写。撰写时建议参考直接下载模型的例子。直接下载模型的modelfile，可以使用

```
OLLMA_HOST=0.0.0.0:your_port /your_path_to_ollama/ollma show model_name --
modelfile
```

来查看。一个例子如下：

```
(base) ➜  ~ OLLAMA_HOST=0.0.0.0:9997 ~/LLM/ollama-linux-amd64 show yi:34b-chat-q6_K --modelfile
# Modelfile generated by "ollama show"
# To build a new Modelfile based on this one, replace the FROM line with:
# FROM yi:34b-chat-q6_K

FROM /home/dell533/.ollama/models/blobs/sha256:243d4aa931101d4765566da2f8b1d3861657c922064b7349d0a8e7210ecd852e
TEMPLATE """<|im_start|>system
{{ .System }}<|im_end|>
<|im_start|>user
{{ .Prompt }}<|im_end|>
<|im_start|>assistant
"""
PARAMETER num_ctx 4096
PARAMETER stop "<|im_start|>"
PARAMETER stop "<|im_end|>"
(base) ➜  ~
```

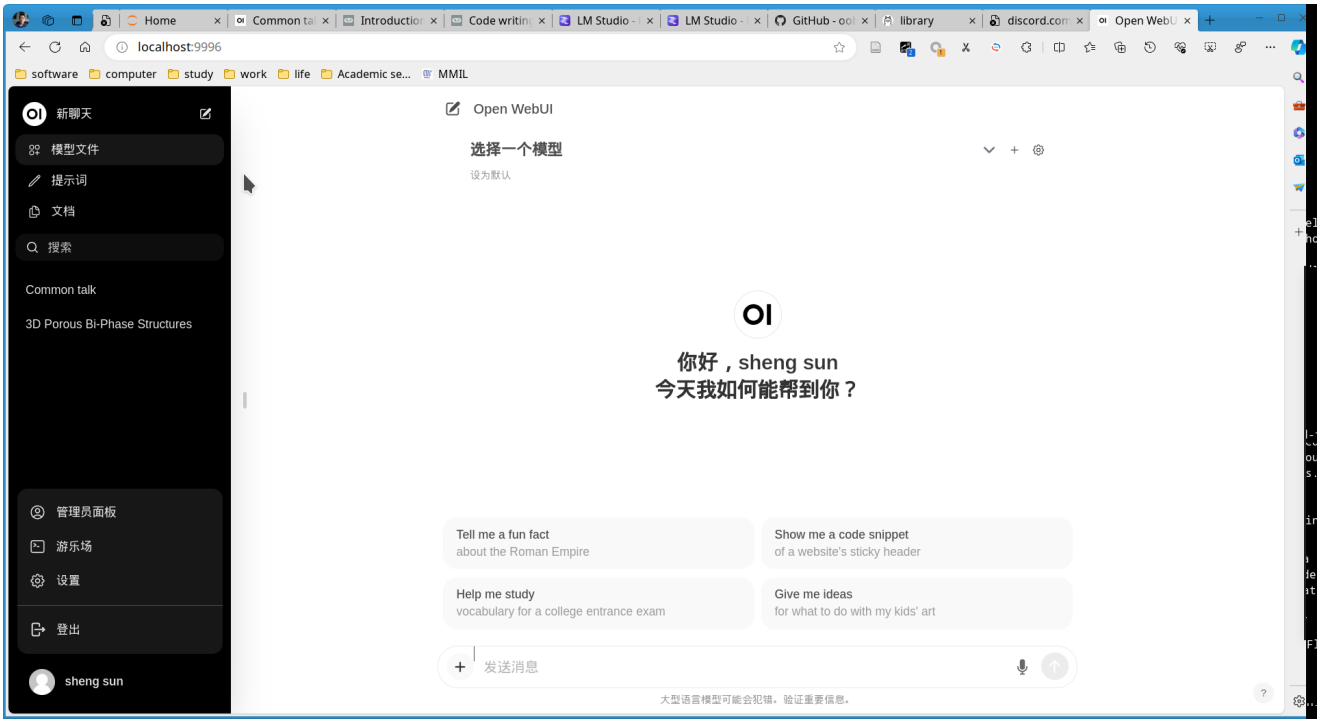
相比于lm-studio和text-generation-webui启动的API服务，OLLAMA启动的服务，可以允许用户在各自的客户端启动不同的模型。而lm-studio和text-generation-webui的模型在服务器端就要选定了。也就是说，lm-studio和text-generation-webui的API可以在客户端使用的时候不指定模型，不会报错，而OLLAMA的客户端一定要指定具体模型。这也给我们带来了方便，比如说，笔者开发的一个使用RAG和Agent的代码自动生成程序，可以调用不同的LLM互相进行代码检查和纠错，并提供新的代码。

# 几种使用方式和推荐程序

## 聊天窗口方式

推荐两个程序：open-webui 和 anythingLLM。

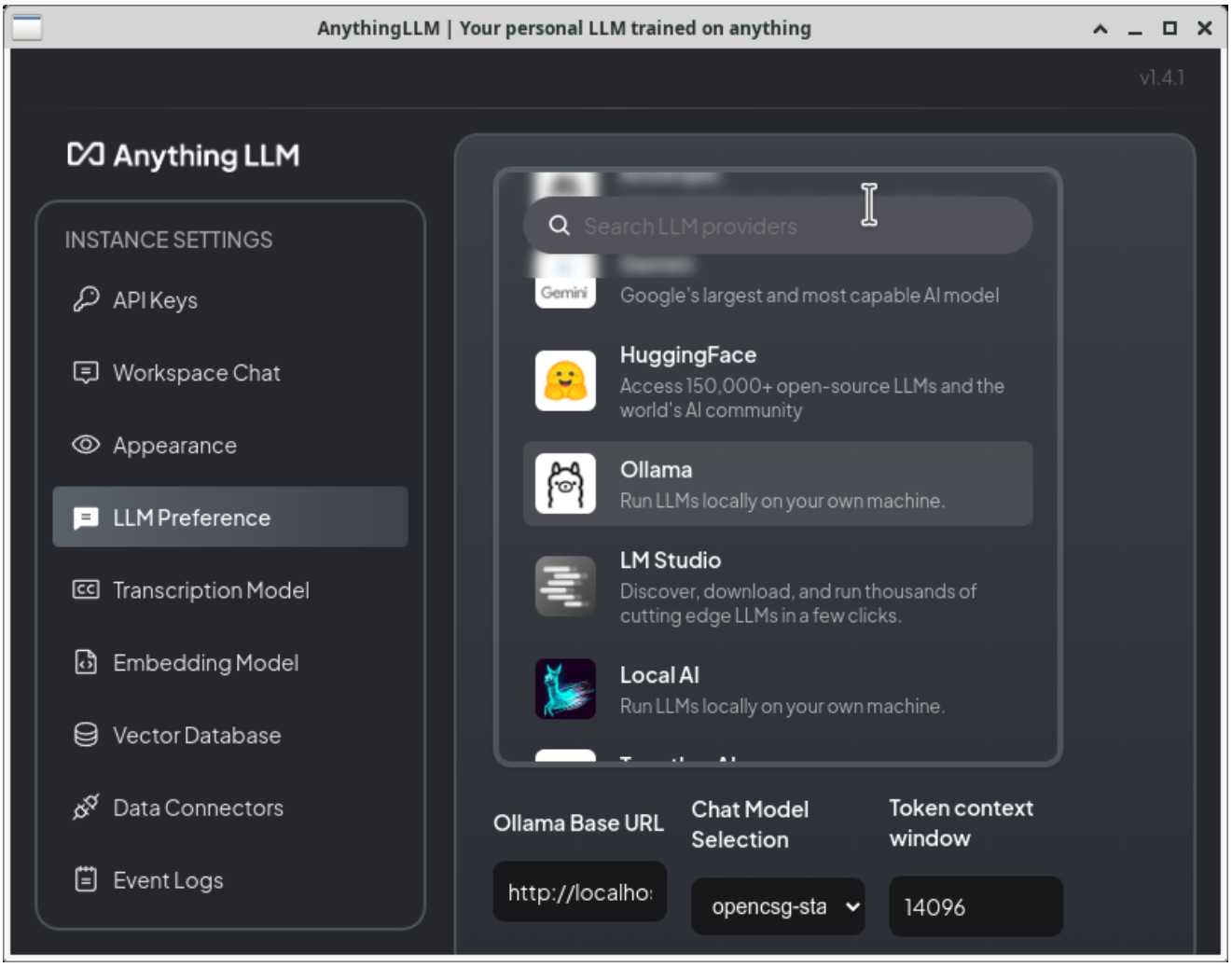
Open WebUI是一个开源的浏览器界面框架，可以使用docker运行，也可以直接下载到本地，进行本地运行。在运行前，要设置好OLLAMA serve的端口和openWebUI的打开端口。




AnythingLLM使用的技术是RAG，检索增强产生。在每个对话session中，用户可以上传pdf等文档或提供网页链接。AnythingLLM在回答问题之前，会先根据问题去查询文档和搜索网页，

把查到的内容和问题一起提交给大模型，从而使大模型产生更精准的回答。

AnythingLLM的设置页面：




AnythingLLM的提问页面：

Anything LLM

+ New Workspace

RefRd

AI4R

MECH

default

+ New Thread

ELECTROCHEM

```python
# Assign pixel values according to pore size distribution
image = (pore_size_max - pore_size_min) * image + pore_size_min
```

Copy code

06. Convert the grayscale image into a TIFF format file:

```python
# Save the image as a TIFF file
img = Image.fromarray(image.astype('uint8'))
img.save('porous_material.tiff')
```

Copy code

This code generates a random 3D porous material with a specified pore density, minimum and maximum pore sizes. You can adjust the parameters to create different types of porous materials.

*Show Citations* >

tell me about the design of metamaterials

Metamaterials are artificial composite materials designed to exhibit unique electromagnetic, mechanical, acoustic or thermal properties that cannot be achieved by naturally occurring materials. They have been widely studied in recent years due to their potential applications in fields such as wave manipulation and energy absorption.

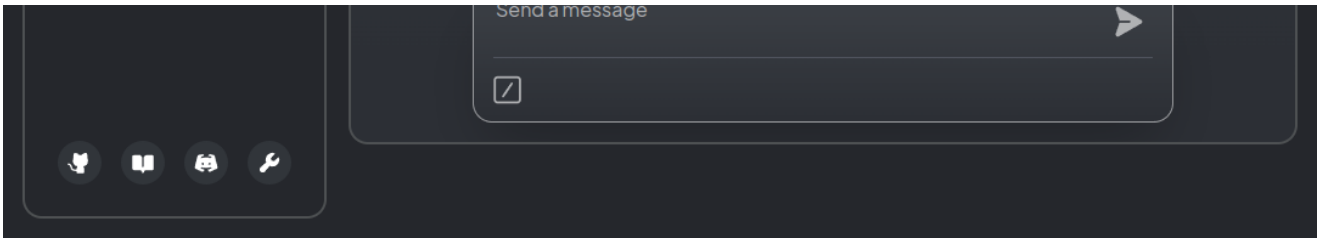
The design of metamaterials involves the optimization of a specific property, which can be achieved through various computational methods. Some popular techniques include topology optimization (e.g., based on the Solid Isotropic Material with Penalization method) and inverse design approaches. Topology optimization aims to find an optimal arrangement of material components in a domain to achieve a desired property, while inverse design focuses on reversing-engineering a metamaterial's design to obtain specific properties.

Despite the availability of computational tools for designing metamaterials, traditional trial and error methods have limitations due to the vastness of the design space. Therefore, machine learning (ML) approaches have emerged as promising alternatives for accelerating the design process and identifying new materials with desired properties.

ML-based models can be trained to predict material properties based on the design parameters or simulate complex processes such as wave propagation in metamaterials. However, developing accurate and generalizable ML models requires a large amount of data and computational resources. This presents challenges when designing metamaterials with novel structures or properties, where traditional methods may still be more effective.

In summary, the design of metamaterials involves optimizing specific material properties while considering factors such as fabrication cost, manufacturability, and ease of application in practical applications. ML-based computational tools can play a significant role in accelerating this process but also have limitations when dealing with complex materials or properties that are not well understood by the community.

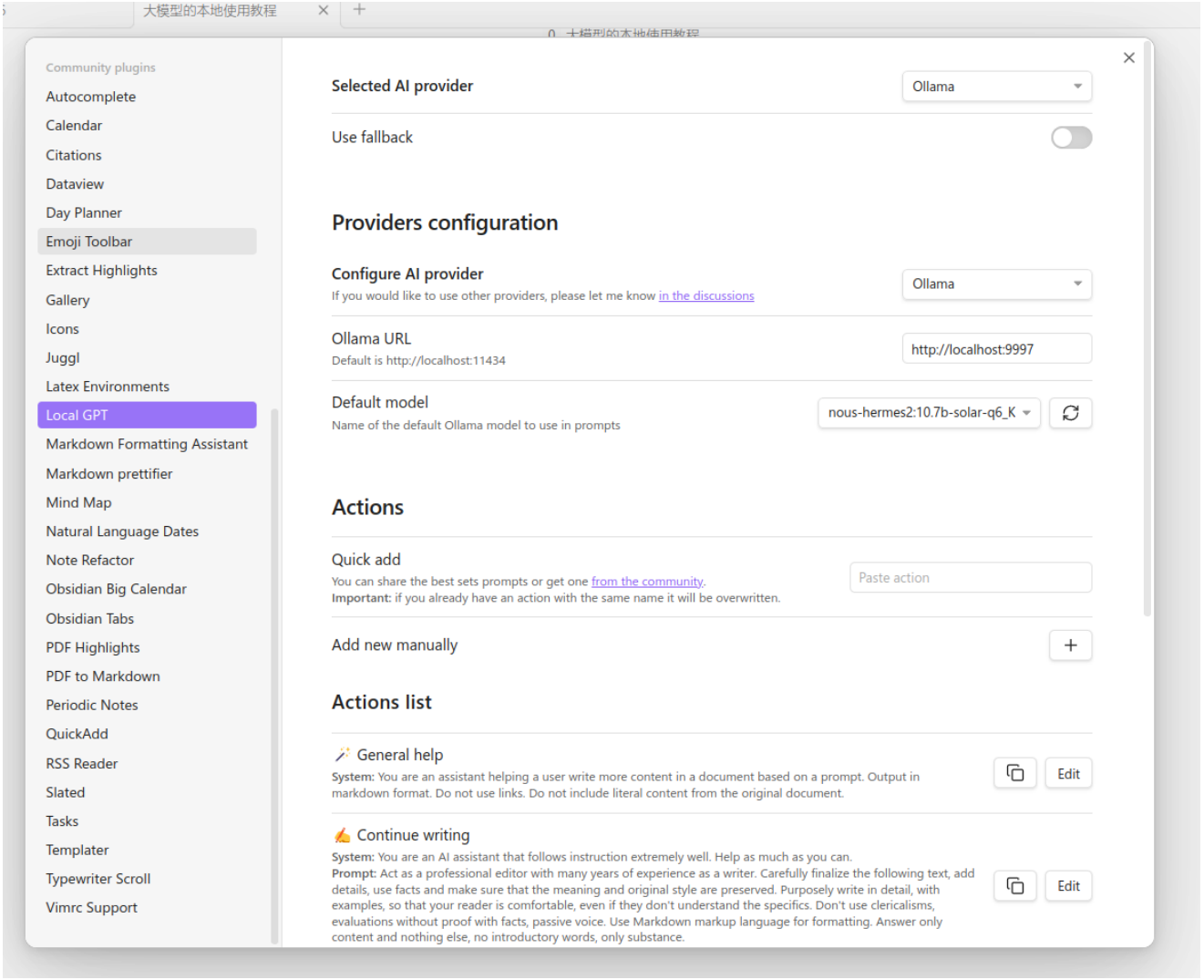
*Show Citations* >




## 写作方式

Obsidian是笔者认为目前最好的卡片式笔记软件。Obsidian有许多的插件，Local GPT插件可以使得在obsidian中直接使用OLLAMA API的模型。

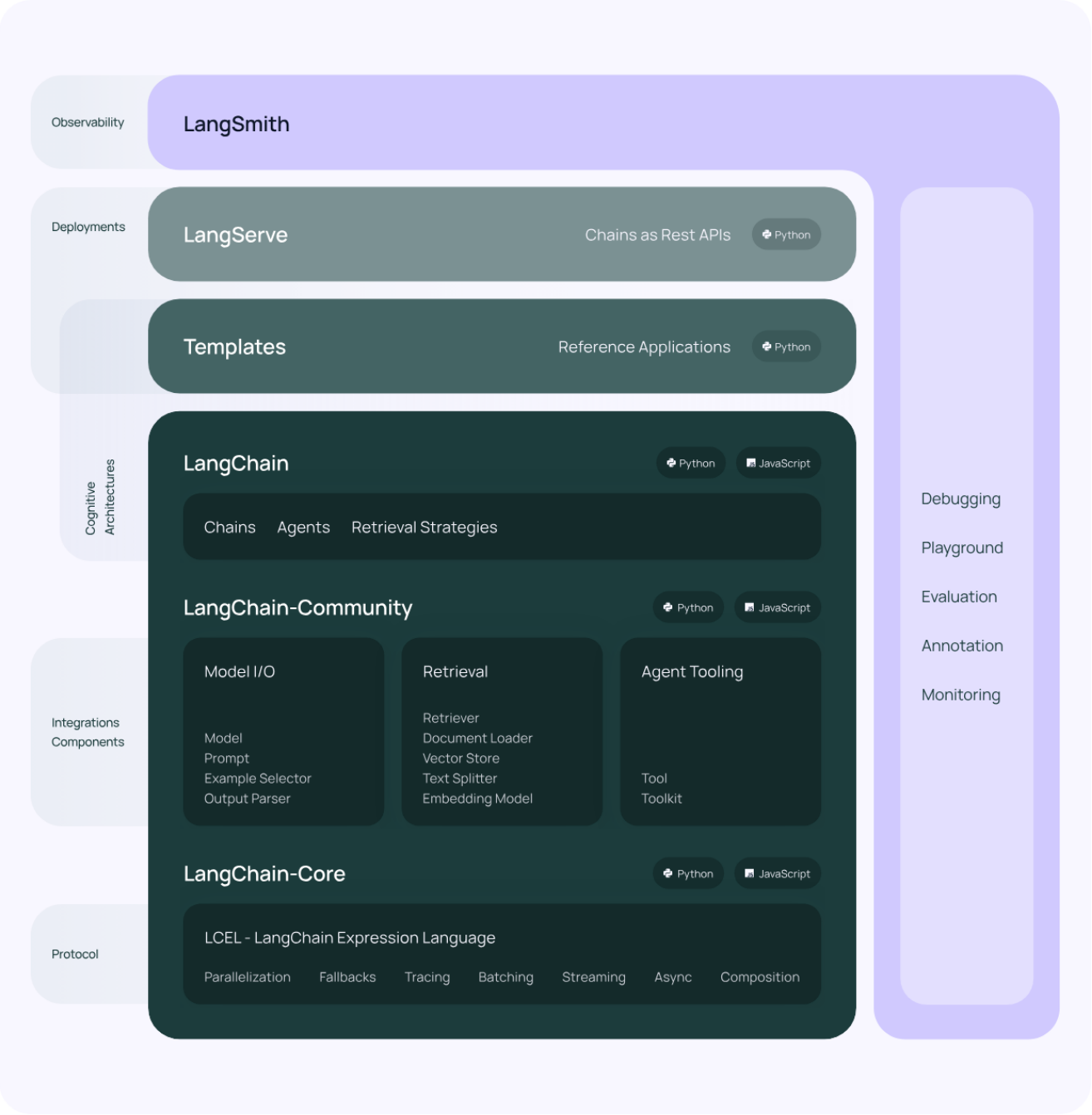
Obsidian的local GPT插件中的API设置页面：




## 编程方式

基于大模型的软件应用，以后必然会充斥于社会的各个方面。目前已经有若干面向大模型应用的程序编码框架，当前最受欢迎的是langchain。它提供了python和javascript两种编程语言的兼容包，有模型IO、Retrieva、Agents、Chains、Memory和Callbacks模块，覆盖了当前最火热的大模型应用技术RAG和Multi-Agent。这两个技术我在后面的文章中会进行介绍。

Langchain的架构图：




笔者使用langchain开发的使用RAG和Agent的代码自动生成程序，可以调用两个不同的LLM互相进行代码检查和纠错，并提供新的代码。

```
        # Prompt
        prompt = PromptTemplate(
            template=template,
            input_variables=["context", "question", "generation", "error"],
            partial_variables={"format_instructions": parser.get_format_instructions()}
        )

        ## Try to let models to chat with each other.
        iter = state_dict["iterations"]
        if iter%2 == 0:
            model = ChatOpenAI(openai_api_base = "http://localhost:9997/v1",
                        openai_api_key = "sk-11111111111111111111111",
                        temperature=0.2, model="dolphin-2.7-mixtral-8x7b",
                        streaming=True, max_tokens=12000)
        else:
            model = ChatOpenAI(openai_api_base = "http://localhost:9997/v1",
                        openai_api_key = "sk-11111111111111111111111",
                        temperature=0.2, model="dolphincoder",
                        streaming=True, max_tokens=12000)
        """
        else:
            model = ChatOpenAI(openai_api_base = "http://localhost:9997/v1",
                        openai_api_key = "sk-11111111111111111111111",
                        temperature=0.2, model="phind-codellama-34b",
                        streaming=True, max_tokens=12000)
        """
        llm_wo_tool =  model
```

至此，相信大家能够初步进行自己的大模型服务搭建和应用了。