Ollma部署llama3

https://github.com/ollama/ollama

Ollama 是一个开源框架，旨在帮助用户在其本地计算机上轻松地管理和部署大型语言模型（LLM）。

它提供了一个轻量级且可扩展的解决方案，支持多种开源大模型，包括但不限于 Llama 、Gemma、Mistral 等，并允许用户自定义和创建自己的模型。

Ollama 的主要特点包括：

1. **本地化部署**：允许企业或个人在本地环境中部署大型语言模型，提高计算效率并降低延迟。
2. **模型管理**：Ollama 支持多种大型语言模型，用户可以根据需要选择和安装不同的模型。
3. **Open-WebUI集成**：Ollama 可以与 Open-WebUI 快速集成，后者是一个开源的 Web 界面，用于与大型语言模型进行交互。
4. **命令行交互**：Ollama 提供了一个基于命令行的交互界面，用户可以输入指令（prompt）并获取模型生成的响应。
5. **优化与微调**：Ollama 允许用户对模型进行优化，包括微调和强化学习，以提高模型在特定任务上的性能。

部署 Ollama 的基本步骤通常包括：

- 安装 Ollama。
- 选择并安装所需的大型语言模型。
- 安装并配置 Open-WebUI（如果需要）。
- 测试语言模型的效果并进行必要的优化。

以下是Ollama部署llama3操作的演示

/mnt/workspace下执行

```
conda create -n ollama python=3.10
conda activate ollama
export OLLAMA_MODELS=/mnt/workspace/models
curl -fsSL https://ollama.com/install.sh | sh
```

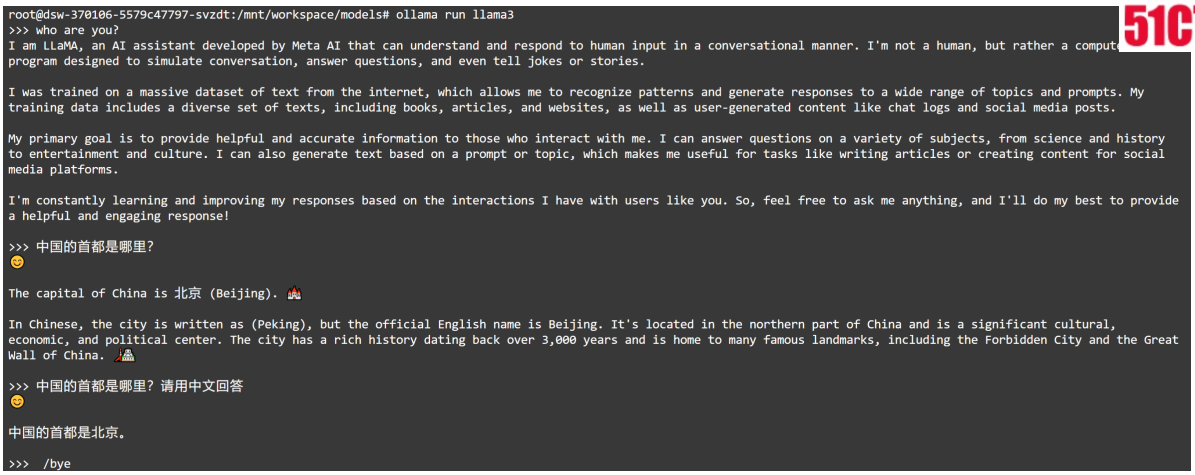
安装成功后执行

```
ollama serve
```

另外启动一个终端执行

```
ollama --version

ollama run llama3
```



REST API

Ollama has a REST API for running and managing models.

Generate a response

```
curl http://localhost:11434/api/generate -d '{
  "model": "llama3",
  "prompt":"Why is the sky blue?"
}'
```

If `stream` is set to `false`, the response will be a single JSON object:

```
curl http://localhost:11434/api/generate -d '{
  "model": "llama3",
  "prompt": "Why is the sky blue?",
  "stream": false
}'
```

在 `curl` 命令中，向一个本地服务器发送一个 HTTP POST 请求，请求的目的是生成文本。

这个请求包含了几个参数：

- `model`：指定要使用的模型，这里是 "llama3"。
- `prompt`：提供了一个提示（prompt），在这个例子中是 "Why is the sky blue?"，模型将基于这个提示生成文本。
- `stream`：这个参数决定了响应的输出方式。

当 `stream` 设为 `true` 时：

- 服务器会立即开始发送响应数据，即使整个响应还没有完全生成完毕。

  这意味着客户端可以逐步接收到生成的文本，而不必等待整个生成过程完成。

  这种"流式"输出对于需要实时查看生成内容的场景很有用。

当 `stream` 设为 `false` 时：

- 客户端会等待服务器完成整个文本生成过程后才接收到响应。

  这意味着服务器会在内部完全处理完生成任务后，一次性将所有生成的文本发送给客户端。

  这种方式适用于需要整个生成文本块的场景。

简而言之，`stream`参数的设置决定了客户端是希望实时接收生成内容（`true`），还是希望在生成后一次性接收全部内容（`false`）。选择哪种方式取决于客户端的应用场景和需求。

Chat with a model

```
curl http://localhost:11434/api/chat -d '{
  "model": "llama3",
  "messages": [
    { "role": "user", "content": "why is the sky blue?" }
  ],
  "stream": false
}'
```

上述curl命令都是在与Ollama的REST API进行交互, 但它们调用的端点(endpoint)和请求参数有所不同, 服务于不同的使用场景。

1. `/api/generate`端点:
    - 该端点用于生成单次回复。
    - 请求参数:
        - `model`: 指定要使用的模型, 这里是"llama3"。
        - `prompt`: 提供一个prompt, 模型将基于这个prompt生成回复。
    - 适用场景: 当你只需要针对某个prompt生成一个回复, 而不需要长期的多轮对话时, 可以使用这个端点。
2. `/api/chat`端点:

- 该端点用于进行多轮对话。
    - 请求参数:
        - `model`: 指定要使用的模型, 这里也是"llama3"。
        - `messages`: 提供一个消息列表, 每个消息都有一个角色(如"user")和内容。模型将基于这些消息的上下文生成回复。
    - 适用场景: 当你需要与模型进行多轮对话, 保持对话的上下文时, 应该使用这个端点。这在实现聊天机器人等应用时非常有用。

简而言之, `/api/generate`用于单次回复的生成, 而`/api/chat`用于多轮对话。它们代表了与语言模型交互的两种主要方式。

选择哪个端点取决于你的具体使用场景。

如果你只需要一次性的回复,`/api/generate`更简单直接。如果你要构建一个聊天应用,需要长期的多轮对话,则`/api/chat`更适合。

上面是一个对话模型生成助手回复的过程记录。

上面给出的curl命令只包含了一条用户消息, 还没有充分体现多轮对话。

为了展示多轮对话, 我们可以在`messages`参数中传入更多的消息, 包括用户消息和助手消息。

下面是一个修改后的示例:

```
curl http://localhost:11434/api/chat -d '{
  "model": "llama3",
  "messages": [
    { "role": "user", "content": "Why is the sky blue?" },
    { "role": "assistant", "content": "The sky appears blue due to a phenomenon
called Rayleigh scattering. This occurs when sunlight enters Earth's atmosphere
and interacts with molecules of gases like nitrogen and oxygen." },
    { "role": "user", "content": "Can you explain more about how Rayleigh
scattering works?" }
  ],
  "stream": false
}'
```

在这个例子中:

1. 第一条消息是用户询问"天空为什么是蓝色的"。
2. 第二条消息是助手的初步回复, 解释了瑞利散射现象。
3. 第三条消息是用户要求助手进一步解释瑞利散射的工作原理。

当发送这个请求时, 模型会考虑之前的所有消息, 然后生成一个新的助手回复, 进一步阐述瑞利散射。这样就形成了一个多轮对话的流程。

你可以继续添加更多的用户和助手消息, 模拟一个完整的对话过程。每次请求, 模型都会基于所有历史消息生成回复, 保持对话的连贯性。

这种多轮对话的交互方式更接近真实的聊天场景, 因为它保留了对话的上下文。模型可以根据之前的信息来生成更加相关和自然的回复。

在命令行中, 如果命令太长或包含特殊字符, 直接执行可能会出现问题。 上面的修改后的示例的curl命令没有返回回复。

让我们尝试另一种方法: 将JSON请求体保存到一个单独的文件中, 然后在curl命令中引用这个文件。

首先, 创建一个名为`request.json`的文件, 并将JSON请求体复制进去:

```
{
  "model": "llama3",
  "messages": [
    { "role": "user", "content": "Why is the sky blue?" },
    { "role": "assistant", "content": "The sky appears blue due to a phenomenon
called Rayleigh scattering. This occurs when sunlight enters Earth's atmosphere
and interacts with molecules of gases like nitrogen and oxygen." },
    { "role": "user", "content": "Can you explain more about how Rayleigh
scattering works?" }
  ],
  "stream": false
}
```

确保JSON格式正确, 并将文件保存。

然后, 修改curl命令, 使其引用这个文件:

```
curl -X POST -H "Content-Type: application/json" -d @request.json
http://localhost:11434/api/chat
```

这个命令使用了以下选项:

- `-X POST`: 指定HTTP方法为POST。
- `-H "Content-Type: application/json"`: 设置请求头, 告诉服务器我们发送的是JSON格式的数据。
- `-d @request.json`: 读取 `request.json` 文件的内容作为请求体。

现在, 尝试执行这个修改后的curl命令。它应该可以正常工作, 向ollama的 `/api/chat` 端点发送请求, 并接收到模型生成的回复。