拥抱大模型：大模型 API 调用、本地配置及相关应用

龚舒凯

2025 年 3 月 3 日

讲座目录

个人简介

龚舒凯，2022 级应用经济-数据科学双学位实验班学生。

- 有 CV、NLP、AI4Science、多模态理解/生成等领域项目经验。
- 国赛省一 (2023)、美赛 M 奖 (2023)。
- 在数据挖掘国际顶会 WWW 2025 上以共同一作身份发表论文。
- 曾在直观医疗研发部担任算法实习生。

大模型 API 的使用

什么是大模型 API (Application Programming Interface)?

- 可以将大模型 API 看作是一个可以**接受输入并返回输出**的函数 f。

$$\text{output} = f(\text{input})$$

- 这个 f 不在本地，而是在远程服务器（比如 OpenAI 的服务器、华为云服务器、阿里云等等）上。当我们使用 API 的时候，我们实际上是在**远程调用**这个函数。

大模型 API 的使用

大模型 API 和我们平时用的大模型区别是什么？

- 我们平时用的大模型都是**图形化界面**。换句话说，我们登上http://deepseek.com，http://chatgpt.com这些网页，然后在网页上输入文本，然后点击“发送”按钮，然后等待网页返回结果。
- 大模型 API 往往是通过**代码调用**的。我们需要写好一段 API 的调用代码，写好一段 prompt（也就是我们的输入），然后运行这段代码，然后等待 API 返回结果。

大模型 API 的使用

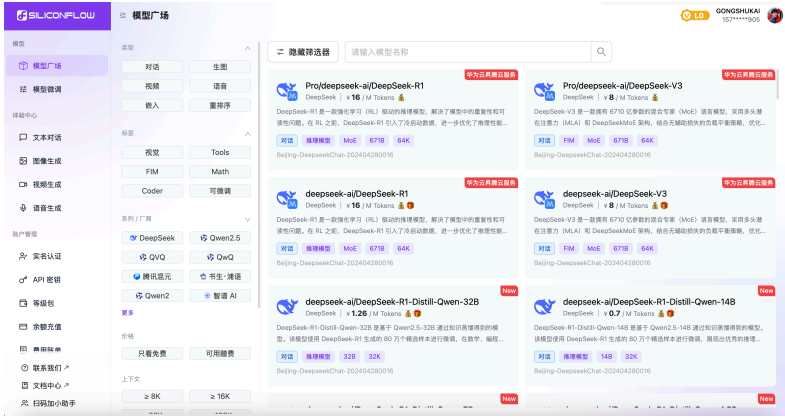
我们为什么要用大模型 API?

- **高并发:** 简单来说，如果你用的网页版 deepseek，你只能一条条问问题。如果你用 API，你可以同一时间发送几千条问题。
- **灵活:** 你可以设置大模型输出的一些“特性”。比如灵活程度 (temperature)、最大 token 数 (max token)、最大生成长度 (max length) 等等。

大模型 API 的使用

怎么用大模型 API?

1. **注册一个可以调用 API 的账号:** 硅基流动https://cloud.siliconflow.cn/models，新用户注册送 14 元额度，每邀请一个好友送 14 元额度。



大模型 API 的使用

怎么用大模型 API?

1. **注册一个可以调用 API 的账号:** 硅基流动https://cloud.siliconflow.cn/models，新用户注册送 14 元额度，每邀请一个好友送 14 元额度。
2. **编写调用 API 的函数:** 可以根据自己需要进行调整，可以让 GPT 帮你写代码脚本。以下展示一个示例。

```python
async def send_message_to_api(semaphore, client: AsyncOpenAI, index, feedback, text_prompt, args):
    messages = [{
        "role": "user",
        "content": [{"type": "text","text": feedback,},
                    {"type": "text","text": text_prompt,}]}
    ]
    while True:
        async with semaphore:
            try:
                response = await client.chat.completions.create(
                        model=args.model,
                        messages=messages,
                        stream=False
                    )
                print(f"Finish requesting for No.{index} feedback. Response: {response.choices[0].message.content}")
                return response.choices[0].message.content
            except RateLimitError:
                print(f"Rate limit exceeded for No.{index}. Retrying in 40 secs...")
                await asyncio.sleep(40)
            except Exception as e:
                print(f"An error occurred for No.{index}: {e}")
                return []
```

大模型 API 的使用

怎么用大模型 API?

1. **注册一个可以调用 API 的账号:** 硅基流动https://cloud.siliconflow.cn/models，新用户注册送 14 元额度，每邀请一个好友送 14 元额度。
2. **编写调用 API 的函数:** 可以根据自己需要进行调整，可以让 GPT 帮你写代码脚本。以下展示一个示例。

```python
async def process_texts_in_folder(data, output_csv, text_prompt, max_concurrent_requests, args):

    # Using the AsyncOpenAI client
    client = AsyncOpenAI(
        api_key=args.api,
        base_url=args.url,
    )

    # Define the maximum number of concurrent requests
    semaphore = asyncio.Semaphore(max_concurrent_requests)
    tasks = []
    for index, row in data.iterrows():
        feedback = row['conclusion_clean']
        keywords = row['keywords']
        if feedback:
            tasks.append(asyncio.create_task(send_message_to_api(semaphore, client, index, feedback, text_prompt, args)))
            # No need to sleep here as concurrency is controlled by semaphore

    # Do all tasks and collect results in `responses`
    responses = await asyncio.gather(*tasks)
    return responses
```

大模型 API 的使用：案例解析

IS 是一家出售手术机器人的公司。它的手术机器人会得到大量的售后反馈。

- 一般而言，售后信息可以被分为 10 类，比如“缺少零件”、“配件损坏”、“无法连接网络”等等。
- 现在公司有一个大型的 csv 文件，每一行都是一条售后信息。每一条售后信息都有将近 2000 字。
- **你的任务：把每一条售后信息都标注一个标签，对应 10 类问题里的一种。**

这个任务非常适合使用大模型 API 标注，因为：

- 现有的大模型有着非常强大的基座能力，可以很好地理解文本。
- LLM API 支持异步并发。如果设置并发量为 300，就可以一次性标注 300 条售后信息的标签。

大模型 API 的使用

使用 Majority Voting（群体投票）这一技巧：

- 可以使用 Claude，GPT-4o，Deepseek-V3，Kimi-1.5，Llama-3.3，Llama（Deepseek 蒸馏版），Qwen2.5 等等多个大模型 API 处理一条售后信息。
- 根据每个大模型的投票结果，决定最终的标签是什么。
- 这些大模型的能力需要有所区分，比如 Claude，GPT-4o，Deepseek-V3，Llama-3.3，Qwen2.5 是非推理模型，而 Llama（Deepseek 蒸馏版）是推理模型。

feedback	keyword	summarizati	label (claude)	label (gpt4o)	label (deepseek-v3)	label (llama-3.3-70b)	label (deepseek-llama)	label (qwen2.5-72b)
it was report	[('shear analy	Cracked or b	0	0	0	0	0	0
it was report	[('shear analy	Cracked or b	0	0	0	0	0	0
it was report	[('shear analy	Cracked or b	0	0	0	0	0	0
an investigat	[('manual rot	Roll gear fract	6	6	6	6	6	6

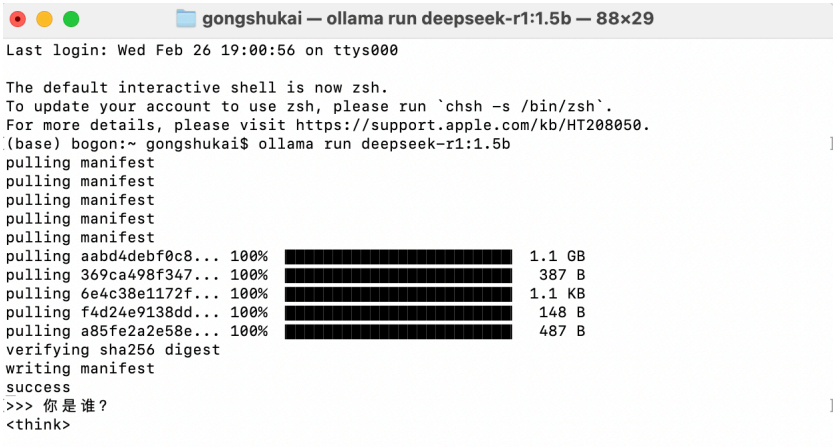
构建本地 chatbot

- 大模型 API：优点是高并发、灵活，缺点是需要网络连接，而且由于部署在云服务器，可能会有信息泄露的问题。
- **大模型本地配置：不需要网络连接，而且数据都在本地，不会泄漏！**

构建本地 chatbot

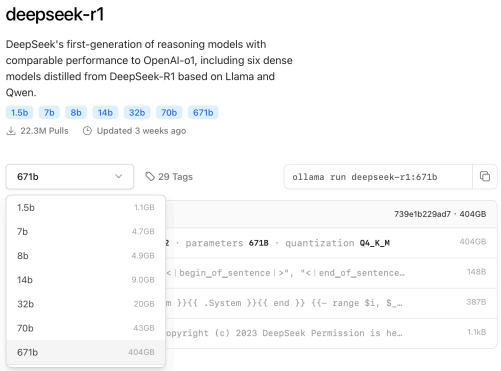
一步就能到位的方法：以 MacOS 为例 (Windows 同理)

- 安装 ollama: https://ollama.com
- 打开你的终端 (MacOS 上叫 terminal，Windows 上叫 cmd，命令提示行)，运行 `ollama run deepseek-r1:1.5b`



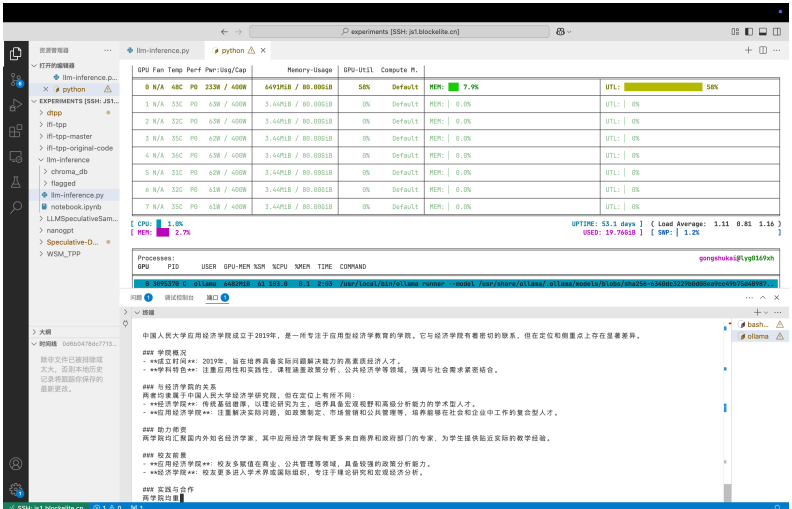
构建本地 chatbot

- 当你运行 `ollama run deepseek-r1:1.5b` 的时候，电脑会下载一个**参数量为 1.5B 的模型**的模型，然后在你的电脑上运行这个模型。你可以在终端上输入问题，然后这个模型会返回答案。
- **使用满血版 Deepseek-r1 不现实：** 671B 参数量的模型占用内存非常大，本地计算机不可能运行。即便运行起来了，受限于内存带宽，输出速度非常慢。



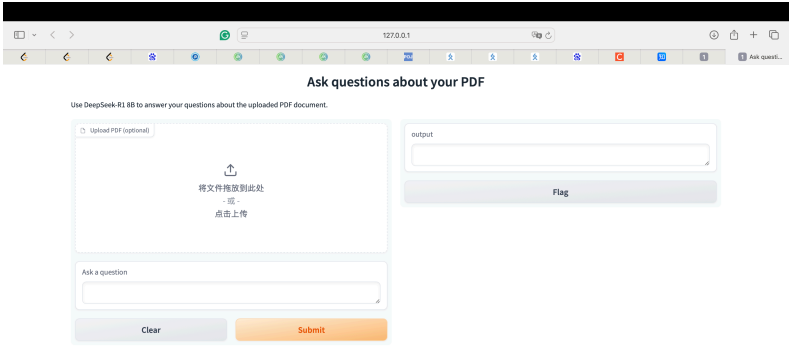
构建本地 chatbot

- 如果你想要运行更加强大的模型 (例如 8B 的、14B 的、32B 的模型)，你需要租赁一台深度学习**服务器** (例如，有一张 24G 显存 NVIDIA 3090 GPU)。



构建本地文献阅读小助手

- 你可能已经习惯于把一个 PDF 扔给 Deepseek，让它帮你读帮你概括。
- 可以离线实现吗？怎么本地配置一个帮你读论文的大模型？



检索增强生成 (RAG)

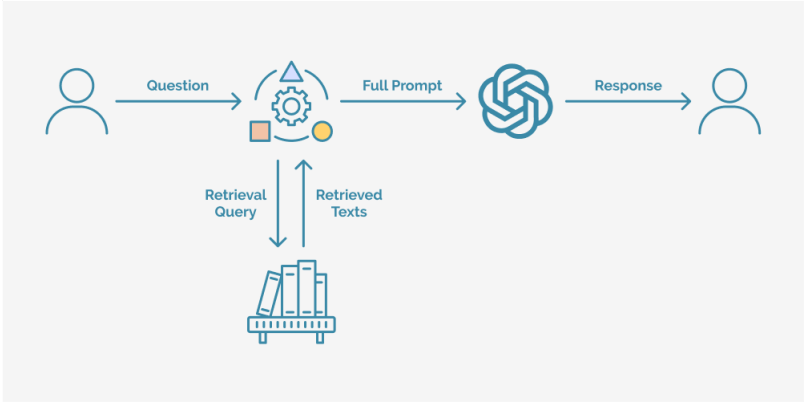


图 1: 检索增强生成 (RAG)

构建本地文献阅读小助手

如何用 Python 实现?

- `import ollama`：ollama 库 ⇒ 本地运行大模型。
- `import gradio as gr` ⇒ 构建交互式界面。
- `langchain` 库 ⇒ 检索增强生成 (RAG)。
  - `from langchain_community.document_loaders import PyMuPDFLoader`：读取 PDF 文档。
  - `from langchain.text_splitter import RecursiveCharacterTextSplitter`：文本分割器，将长文本切分成更小的片段。
  - `from langchain.vectorstores import Chroma`：向量数据库，用于存储和检索文本嵌入向量。
  - `from langchain_community.embeddings import OllamaEmbeddings`：用 Ollama 生成文本嵌入向量。

RAG 核心组件

- 处理 PDF& 向本地大模型提问。

```
def ask_question(pdf_bytes, question):
    text_splitter, vectorstore, retriever = process_pdf(pdf_bytes) # Process the PDF
    if text_splitter is None:
        return None  # No PDF uploaded
    result = rag_chain(question, text_splitter, vectorstore, retriever) # Return the results with
    return {result}
```

RAG 核心组件

- 从 PDF 中抽取和问题强相关的信息，发送给本地大模型。

```python
def rag_chain(question, text_splitter, vectorstore, retriever):
    """
    This function takes as input:
        - The question we want to ask the model
        - The text_splitter object to split the PDF and read into chunks
        - The vectorstore for retrieving embeddings
        - The retriever objects which retrieves data from the vectorstore
    """
    retrieved_docs = retriever.invoke(question) # In this step, we will find the part of the docum
    formatted_content = combine_docs(retrieved_docs) # We will then combine the retrieved parts of
    return ollama_llm(question, formatted_content) # Run the model on the question, and the releva
```

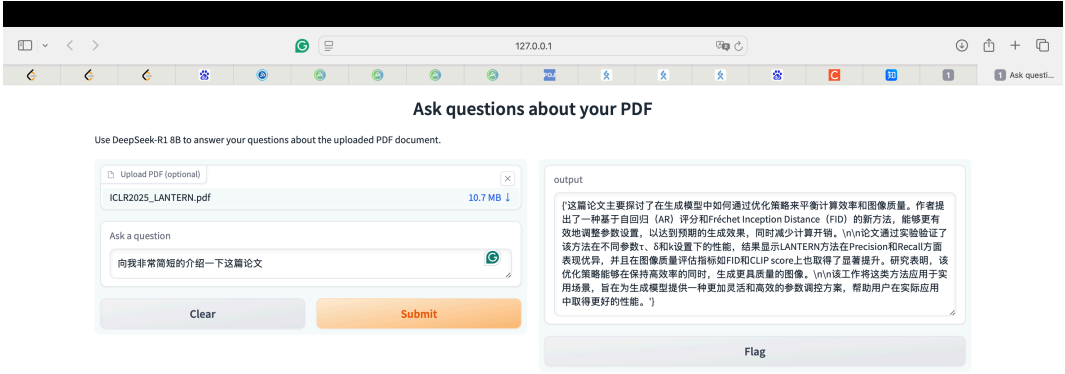
RAG 核心组件

- 本地大模型生成答复。

```python
def ollama_llm(question, context):

    # Format the prompt with the question and context to provide structured input for the AI
    formatted_prompt = f"Question: {question}\n\nContext: {context}"
    # Send the structured prompt to the Ollama model for processing
    response = ollama.chat(
        model="deepseek-r1:8b",  # Specifies the AI model to use
        messages=[{'role': 'user', 'content': formatted_prompt}]  # Formats the user input
    )
    # Extract the AI-generated response content
    response_content = response['message']['content']
    # Remove content inside <think>...</think> tags to clean up AI reasoning traces
    final_answer = re.sub(r'<think>.*?</think>', # We're searching for think tags
                          '', # We'll replace them with empty spaces
                          response_content, # In response_content
                          flags=re.DOTALL).strip() # (dot) should match newlines (\n) as well.
    # Return the final cleaned response
    return final_answer
```

构建本地文献阅读小助手



Thank you!