

## 10 | AI连接外部资料库，让Llama Index带你阅读一本书

2023-04-04 徐文浩 来自北京

《AI大模型之美》



你好，我是徐文浩。

有不少人在使用 OpenAI 提供的 GPT 系列模型的时候，都反馈效果并不好。这些反馈中有一大类问题，是回答不了一些简单的问题。比如当我们用中文问 AI 一些事实性的问题，AI 很容易胡编乱造。而当你问它最近发生的新闻事件的时候，它就干脆告诉你它不知道 21 年之后的事情。

本来呢，我写到这里就可以了。不过到了 3 月 24 日，OpenAI 推出了 ChatGPT Plugins 这个功能，可以让 ChatGPT 通过插件的形式链接外部的第三方应用。我自己也还在排队等 waiting list，所以暂时也无法体验。不过，即使有了第三方应用，我们也不能确保自己想要的信息正好被其他人提供了。而且，有些信息和问题我们只想提供给自己公司的内部使用，并不想开放给所有人。这个时候，我们既希望能够利用 OpenAI 的大语言模型的能力，但是又需要这些能力仅仅在我们自己指定的数据上。那么这一讲，就是为了解决这个问题的。

## 大型语言模型的不足之处

我们先来尝试问 ChatGPT 一个人尽皆知的常识，“鲁迅先生去日本学习医学的老师是谁”，结果它给出的答案是鲁迅的好友，内山书店的老板内山完造，而不是大家都学习过的藤野先生。



鲁迅先生去日本学习医学的老师是谁？



鲁迅先生在日本学习医学的时候，他的老师是名叫内山完造（Uchiyama Kanzo）的日本医生。内山完造是东京大学医学部的教授，他教授了鲁迅先生的解剖学、生理学等课程，并且在医学上给予了鲁迅先生很多指导和帮助。鲁迅先生对内山完造非常尊敬，并在日后的文章中多次提到了他。



之所以会出现这样的情况，和大模型的原理以及它使用训练的数据集是有关的。大语言模型的原理，就是利用训练样本里面出现的文本的前后关系，通过前面的文本对接下来出现的文本进行概率预测。如果类似的前后文本出现得越多，那么这个概率在训练过程里会收敛到少数正确答案上，回答就准确。如果这样的文本很少，那么训练过程里就会有一定的随机性，对应的答案就容易似是而非。而在 GPT-3 的模型里，虽然整体的训练语料很多，但是中文语料很少。只有不到 1% 的语料是中文的，所以如果问很多中文相关的知识性或者常识性问题，它的回答往往就很扯。

当然，你可以说我们有一个解决办法，就是多找一些高质量的中文语料训练一个新的模型。或者，对于我们想让 AI 能够回答出来的问题，找一些数据。然后利用 OpenAI 提供的“微调”（Fine-tune）接口，在原来的基础上训练一个新模型出来。



上一周曼联队的比赛的比分分别是多少啊？



很抱歉，作为一个语言模型，我没有实时更新的功能，也无法查询最新的体育比分信息。建议您查看相关的体育新闻网站或应用程序以获取最新比分信息。



如果你问一下新闻类的信息，ChatGPT通常会拒绝回答

这样当然是可以的，就是成本有点高。对于上面那个例子来说，只是缺少一些文本数据，还好说。如果是时效性要求比较强的资讯类的信息，就很难这么做。比如，我们想让 AI 告诉我们前一天足球赛的比分，我们不太可能每隔几个小时就单独训练或者微调一下模型，那样干的成本太高了。

## Bing 的解法——先搜索，后提示

不过对这个领域比较关注的朋友可能就要问了。之前微软不是在 Bing 这个搜索引擎里，加上了 ChatGPT 的问答功能吗？效果似乎也还不错，那 Bing 是怎么做到的呢，是因为他们用了更加厉害的语言模型吗？

虽然我并没有什么内幕消息，不了解 Bing 是怎么做的。但是如果是我的话，会用这样一个解决办法——那就是先搜索，后提示（Prompt）。

1. 我们先通过搜索的方式，找到和询问的问题最相关的语料。这个搜索过程中，我们既可以用传统的基于关键词搜索的技术，也可以用 [第 9 讲](#) 我们刚刚介绍过的使用 Embedding 的相似度进行语义搜索的技术。
2. 然后，我们将和问题语义最接近的前几条内容，作为提示语的一部分给到 AI。然后请 AI 参考这些内容，再来回答这个问题。



请你根据下面这些段落，告诉我鲁迅先生在日本学习医学的老师是谁？



我就往仙台10的医学专门学校去。从东京出发，不久便到一处驿站，写道：日暮里。不知怎地，我到现在还记得这名目。其次却只记得水户11了，这是明的遗民朱舜水12先生客死13的地方。仙台是一个市镇，并不大；冬天冷得厉害；还没有中国的学生。

从此就看见许多陌生的先生，听到许多新鲜的讲义。解剖学是两个教授分任的。最初是骨学。其时进来的是一个黑瘦的先生，八字须，戴着眼镜，挟着一迭大大小小的书。一将书放在讲台上，便用了缓慢而很有顿挫的声调，向学生介绍自己道：——

“我就是叫作藤野严九郎的……。”



鲁迅先生在日本学习医学的老师是藤野严九郎。



我在这里，也给了你一个例子的截图。当我们把《藤野先生》里的两个段落给到 AI，然后请 AI 根据这两个段落，回答原来那个问题，就会得到正确的答案，你也可以去看一看。

这也是利用大语言模型的一个常见模式。因为大语言模型其实内含了两种能力。

第一种，是海量的语料中，本身已经包含了的知识信息。比如，我们前面问 AI 鱼香肉丝的做法，它能回答上来就是因为语料里已经有了充足的相关知识。我们一般称之为“世界知识”。

第二种，是根据你输入的内容，理解和推理的能力。这个能力，不需要训练语料里有一样的内容。而是大语言模型本身有“思维能力”，能够进行阅读理解。这个过程里，“知识”不是模型本身提供的，而是我们找出来，临时提供给模型的。如果不提供这个上下文，再问一次模型相同的问题，它还是答不上来的。

## 通过 llama\_index 封装“第二大脑”

我给上面这种先搜索、后提示的方式，取了一个名字，叫做 AI 的“第二大脑”模式。因为这个方法，需要提前把你希望 AI 能够回答的知识，建立一个外部的索引，这个索引就好像 AI

的“第二个大脑”。每次向 AI 提问的时候，它都会先去查询一下这个第二大脑里面的资料，找到相关资料之后，再通过自己的思维能力来回答问题。

实际上，你現在在网上看到的很多读论文、读书回答问题的应用，都是通过这个模式来实现的。那么，现在我们就来自己实现一下这个“第二个大脑”模式。

不过，我们不必从 0 开始写代码。因为这个模式实在太过常用了，所以有人为它写了一个开源 Python 包，叫做 llama-index。那么我们这里，可以直接利用这个软件包，用几行代码来试一试，它能不能回答上鲁迅先生写的《藤野先生》相关的问题。

llama-index 还没有人做好 Conda 下的包，所以即使在 Conda 下还是要通过 pip 来安装。

 复制代码

```
1 pip install llama-index
2 pip install langchain
```


我把从网上找到的《藤野先生》这篇文章变成了一个 txt 文件，放在了 data/mr\_fujino 这个目录下。我们的代码也非常简单，一共没有几行。

 复制代码

```
1 import openai, os
2 from llama_index import GPTVectorStoreIndex, SimpleDirectoryReader
3
4 openai.api_key = os.environ.get("OPENAI_API_KEY")
5
6 documents = SimpleDirectoryReader('./data/mr_fujino').load_data()
7 index = GPTSimpleVectorIndex.from_documents(documents)
8
9 index.save_to_disk('index_mr_fujino.json')
```

输出结果：



 复制代码

```
1 INFO:llama_index.token_counter.token_counter:> [build_index_from_nodes] Total LLM
2 INFO:llama_index.token_counter.token_counter:> [build_index_from_nodes] Total emb
```

注：日志中会打印出来我们通过 Embedding 消耗了多少个 Token。

首先，我们通过一个叫做 SimpleDirectoryReader 的数据加载器，将整个 ./data/mr\_fujino 的目录给加载进来。这里的每一个文件，都会被当成是一篇文档。


然后，我们将所有的文档交给了 GPTSimpleVectorIndex 构建索引。顾名思义，它会把文档分段转换成一个个向量，然后存储成一个索引。

最后，我们会把对应的索引存下来，存储的结果就是一个 json 文件。后面，我们就可以用这个索引来进行相应的问答。

 复制代码

```
1 index = GPTVectorStoreIndex.load_from_disk('index_mr_fujino.json')
2 response = index.query("鲁迅先生在日本学习医学的老师是谁? ")
3 print(response)
```

要进行问答也没有几行代码，我们通过 GPTSimpleVectorIndex 的 load\_from\_disk 函数，可以把刚才生成的索引加载到内存里面来。然后对着 Index 索引调用 Query 函数，就能够获得问题的答案。可以看到，通过外部的索引，我们可以正确地获得问题的答案。

 复制代码

```
1 INFO:llama_index.token_counter.token_counter:> [query] Total LLM token usage: 298
2 INFO:llama_index.token_counter.token_counter:> [query] Total embedding token usag
3
4 鲁迅先生在日本学习医学的老师是藤野严九郎先生。
```

这么一看，似乎问题特别简单，三行代码就搞定了。别着急，我们再看看别的问题它是不是也能答上来？这次我们来试着问问鲁迅先生是在哪里学习医学的。

[📄 复制代码](#)

```
1 response = index.query("鲁迅先生去哪里学的医学? ")
2 print(response)
```

输出结果：

[📄 复制代码](#)

```
1 > Got node text: 藤野先生
2 东京也无非是这样。上野的樱花烂漫的时节，望去确也像绯红的轻云，但花下也少不了成群结队的“清国留学生
3 中国留学生会馆的门房里有几本书买，有时还值得去一转；倘在上午，里面的几间洋房里倒也还可以坐坐的。
4
5 INFO:llama_index.token_counter.token_counter:> [query] Total LLM token usage: 296
6 INFO:llama_index.token_counter.token_counter:> [query] Total embedding token usag
7
8 鲁迅先生去仙台的医学专门学校学习医学。
```

它仍然正确回答了问题。那么，我们搜索到的内容，在这个过程里面是如何提交给 OpenAI 的呢？我们就来看看下面的这段代码就知道了。

[📄 复制代码](#)


```
1 from llama_index import QuestionAnswerPrompt
2 query_str = "鲁迅先生去哪里学的医学? "
3 DEFAULT_TEXT_QA_PROMPT_TMPL = (
4     "Context information is below. \n"
5     "-----\n"
6     "{context_str}"
7     "\n-----\n"
8     "Given the context information and not prior knowledge, "
9     "answer the question: {query_str}\n"
10 )
11 QA_PROMPT = QuestionAnswerPrompt(DEFAULT_TEXT_QA_PROMPT_TMPL)
12
13 response = index.query(query_str, text_qa_template=QA_PROMPT)
14 print(response)
```

这段代码里，我们定义了一个 QA\_PROMPT 的对象，并且为它设计了一个模版。

1. 这个模版的开头，我们告诉 AI，我们为 AI 提供了一些上下文信息（Context information）。
2. 模版里面支持两个变量，一个叫做 context\_str，另一个叫做 query\_str。context\_str 的地方，在实际调用的时候，会被通过 Embedding 相似度找出来的内容填入。而 query\_str 则是会被我们实际提的问题替换掉。
3. 实际提问的时候，我们告诉 AI，只考虑上下文信息，而不要根据自己已经有的先验知识（prior knowledge）来回答问题。


我们就是这样，把搜索找到的相关内容以及问题，组合到一起变成一段提示语，让 AI 能够按照我们的要求来回答问题。那我们再问一次 AI，看看答案是不是没有变。

输出结果：

 复制代码

```
1 鲁迅先生去仙台的医学专门学校学习医学。
```

这一次 AI 还是正确地回答出了鲁迅先生是去仙台的医学专门学校学习的。我们再试一试，问一些不相干的问题，会得到什么答案，比如我们问问红楼梦里林黛玉和贾宝玉的关系。


 复制代码

```
1 QA_PROMPT_TMPL = (  
2     "下面的“我”指的是鲁迅先生 \n"  
3     "-----\n"  
4     "{context_str}"  
5     "\n-----\n"  
6     "根据这些信息，请回答问题: {query_str}\n"  
7     "如果您不知道的话，请回答不知道\n"  
8 )  
9 QA_PROMPT = QuestionAnswerPrompt(QA_PROMPT_TMPL)  
10  
11 response = index.query("请问林黛玉和贾宝玉是什么关系？", text_qa_template=QA_PROMPT)  
12  
13 print(response)
```



输出结果：

```
1 不知道
```

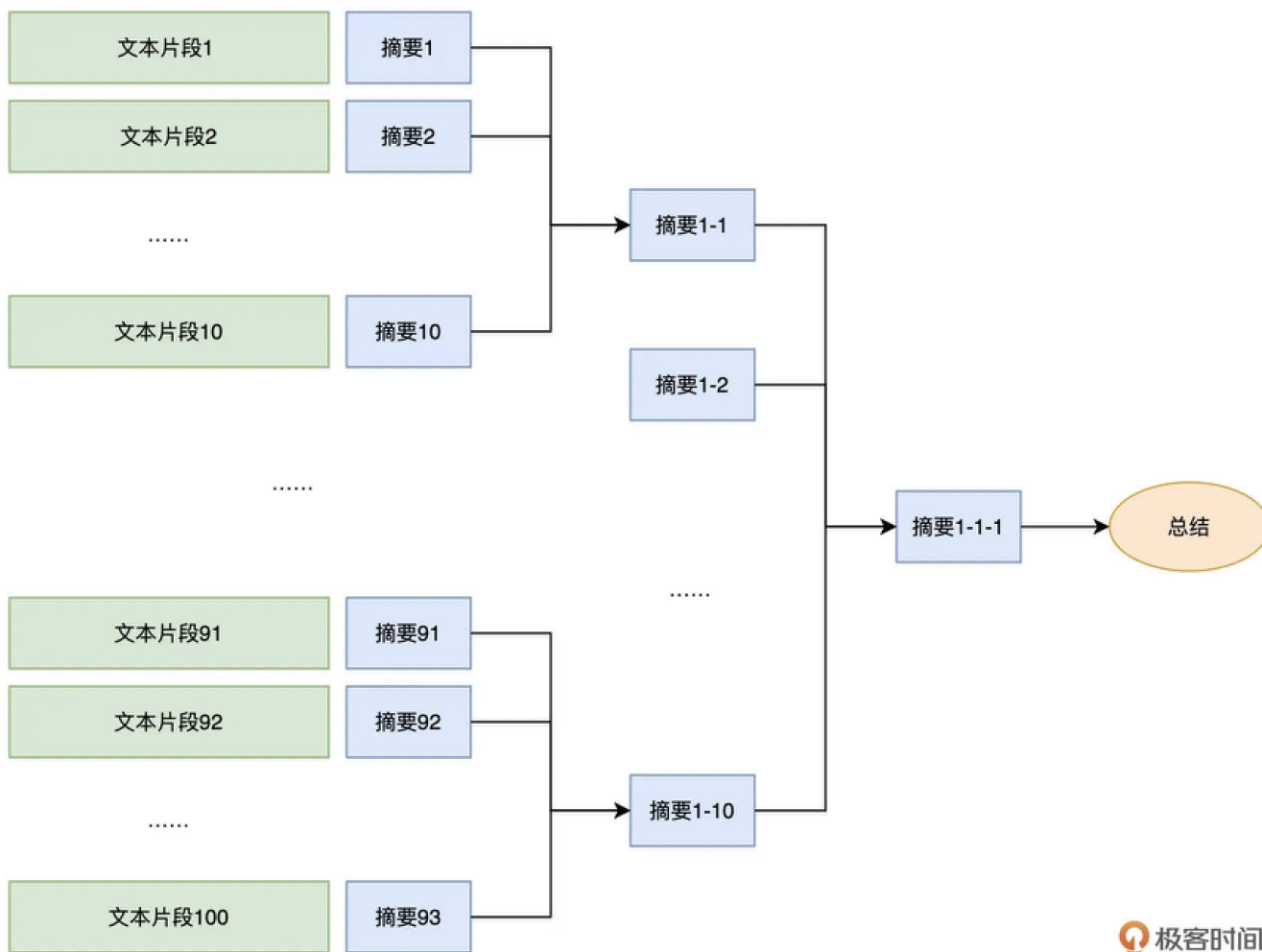
 复制代码

可以看到，AI 的确按照我们的指令回答不知道，而不是胡答一气。

## 通过 llama\_index 对于文章进行小结

还有一个常见的使用 llama-index 这样“第二大脑”的 Python 库的应用场景，就是生成文章的摘要。在前面教你如何进行文本聚类的时候，我们已经看到了可以通过合适的提示语（Prompt）做到这一点。不过，如果要总结一篇论文、甚至是一本书，每次最多只能支持 4096 个 Token 的 API 就不太够用了。

要解决这个问题也并不困难，我们只要进行分段小结，再对总结出来的内容再做一次小结就可以了。我们可以把一篇文章，乃至一本书，构建成一个树状的索引。每一个树里面的节点，就是它的子树下内容的摘要。最后，在整棵树的根节点，得到的就是整篇文章或者整本书的总结了。



通过将文本分片建立树状结构的索引来完成全文的总结

事实上，llama-index 本身就内置了这样的功能。下面我们就来看看要实现这个功能，我们的代码应该怎么写。

首先，我们先来安装一下 spaCy 这个 Python 库，并且下载一下对应的中文分词分句需要的模型。


复制代码

```
1 pip install spacy
2 python -m spacy download zh_core_web_sm
```

接下来的代码很简单，我们选用了 GPTListIndex 这个 llama-index 里最简单的索引结构。不过我们针对自身需求做了两点优化。


首先，在索引里面，我们指定了一个 LLMPredictor，让我们向 OpenAI 发起请求的时候，都使用 ChatGPT 的模型。因为这个模型比较快，也比较便宜。llama-index 默认使用的模型是 text-davinci-003，价格比 gpt-3.5-turbo 要贵上十倍。在我们前面只是简单进行几轮对话的时候，这个价格差异还不明显。而如果你要把几十本书都灌进去，那成本上就会差上不少了。我们在这里，设置了模型输出的内容都在 1024 个 Token 以内，这样可以确保我们的小结不会太长，不会把一大段不相关的内容都合并到一起。

其次，我们定义了使用 SpacyTextSplitter 来进行中文文本的分割。llama-index 默认的设置对于中文的支持和效果都不太好。不过好在它可以让你自定义使用的文本分割方式。我们选用的文章是中文的，里面的标点符号也都是中文的，所以我们就用了中文的语言模型。我们也限制了分割出来的文本段，最长不要超过 2048 个 Token，这些参数都可以根据你实际用来处理的文章内容和属性自己设置。

 复制代码

```
1 from langchain.chat_models import ChatOpenAI
2 from langchain.text_splitter import SpacyTextSplitter
3 from llama_index import GPTListIndex, LLMPredictor, ServiceContext
4 from llama_index.node_parser import SimpleNodeParser
5
6 # define LLM
7 llm_predictor = LLMPredictor(llm=ChatOpenAI(temperature=0, model_name="gpt-3.5-tu
8
9 text_splitter = SpacyTextSplitter(pipeline="zh_core_web_sm", chunk_size = 2048)
10 parser = SimpleNodeParser(text_splitter=text_splitter)
11 documents = SimpleDirectoryReader('./data/mr_fujino').load_data()
12 nodes = parser.get_nodes_from_documents(documents)
13
14 service_context = ServiceContext.from_defaults(llm_predictor=llm_predictor)
15
16 list_index = GPTListIndex(nodes=nodes, service_context=service_context)
```

输出结果：


 复制代码

```
1 WARNING:llama_index.llm_predictor.base:Unknown max input size for gpt-3.5-turbo,
2 INFO:llama_index.token_counter.token_counter:> [build_index_from_nodes] Total LLM
3 INFO:llama_index.token_counter.token_counter:> [build_index_from_nodes] Total emb
```

GPTListIndex 在构建索引的时候，并不会创建 Embedding，所以索引创建的时候很快，也不消耗 Token 数量。它只是根据你设置的索引结构和分割方式，建立了一个 List 的索引。


接着，我们就可以让 AI 帮我们去小结这篇文章了。同样的，提示语本身很重要，所以我们还是强调了文章内容是鲁迅先生以“我”这个第一人称写的。因为我们想要的是按照树状结构进行文章的小结，所以我们设定了一个参数，叫做 `response_mode = "tree_summarize"`。这个参数，就会按照上面我们所说的树状结构把整个文章总结出来。

实际上，它就是将每一段文本分片，都通过 query 内的提示语小结。再对多个小结里的内容，再次通过 query 里的提示语继续小结。

 复制代码

```
1 response = list_index.query("下面鲁迅先生以第一人称‘我’写的内容，请你用中文总结一下:", res
2 print(response)
```

输出结果：

 复制代码


```
1 INFO:llama_index.indices.common_tree.base:> Building index from nodes: 2 chunks
2 INFO:llama_index.token_counter.token_counter:> [query] Total LLM token usage: 978
3 INFO:llama_index.token_counter.token_counter:> [query] Total embedding token usag
4 鲁迅先生回忆了自己在日本学医期间的经历，描述自己在解剖实习中的经历，以及与教授藤野先生的交往。什
```

可以看到，我们只用了几行代码就完成了整个文章的小结，返回的结果整体上来说也还算不错。

## 引入多模态，让 llama-index 能够识别小票

llama\_index 不光能索引文本，很多书里面还有图片、插画这样的信息。llama\_index 一样可以索引起来，供你查询，这也就是所谓的多模态能力。当然，这个能力其实是通过一些多模态的模型，把文本和图片能够联系到一起做到的。在整个课程的第三部分，我们也会专门来看看这些图像的多模态模型是怎么样子的。

这里我们就来看一个 llama\_index [官方样例库](#) 里面给到的例子，也就是把吃饭的小票都拍下来。然后询问哪天吃了什么，花了多少钱。


 复制代码

```
1 from llama_index import SimpleDirectoryReader, GPTVectorStoreIndex
2 from llama_index.readers.file.base import DEFAULT_FILE_EXTRACTOR, ImageParser
3 from llama_index.response.notebook_utils import display_response, display_image
4 from llama_index.indices.query.query_transform.base import ImageOutputQueryTransf
5
6 image_parser = ImageParser(keep_image=True, parse_text=True)
7 file_extractor = DEFAULT_FILE_EXTRACTOR
8 file_extractor.update(
9 {
10     ".jpg": image_parser,
11     ".png": image_parser,
12     ".jpeg": image_parser,
13 })
14
15 # NOTE: we add filename as metadata for all documents
16 filename_fn = lambda filename: {'file_name': filename}
17
18 receipt_reader = SimpleDirectoryReader(
19     input_dir='./data/receipts',
20     file_extractor=file_extractor,
21     file_metadata=filename_fn,
22 )
23 receipt_documents = receipt_reader.load_data()
```

要能够索引图片，我们引入了 ImageParser 这个类，这个类背后，其实是一个基于 OCR 扫描的模型 [Donut](#)。它通过一个视觉的 Encoder 和一个文本的 Decoder，这样任何一个图片能够变成一个一段文本，然后我们再通过 OpenAI 的 Embedding 把这段文本变成了一个向量。


我们仍然只需要使用简单的 SimpleDirectoryReader，我们通过指定 FileExtractor，会把对应的图片通过 ImageParser 解析成为文本，并最终成为向量来用于检索。

然后，我们仍然只需要向我们的索引用自然语言提问，就能找到对应的图片了。在提问的时候，我们专门制定了一个 ImageOutputQueryTransform，主要是为了在输出结果的时候，能够在图片外加上 <img> 的标签方便在 Notebook 里面显示。

 复制代码

```
1 receipts_index = GPTVectorStoreIndex.from_documents(receipt_documents)
2 receipts_response = receipts_index.query(
3     'When was the last time I went to McDonald\'s and how much did I spend. \
4     Also show me the receipt from my visit.',
5     query_transform=ImageOutputQueryTransform(width=400)
6 )
7
8 display_response(receipts_response)
```

输出结果:

 复制代码

```
1 INFO:llama_index.token_counter.token_counter:> [query] Total LLM token usage: 100
2 INFO:llama_index.token_counter.token_counter:> [query] Total embedding token usag
3
4 Final Response: The last time you went to McDonald's was on 03/10/2018 at 07:39:1
```



813  
\*\*\*\* DUPLICATE \*\*\*\*

**\*\* BILLA**



**\*\***

Stony  
16725 Stony Plain Rd  
Edmonton AB T5P 4A9  
Store#: 3859 Tel#: 780-414-8362

Welcome to all day breakfast  
McDonald's

192

KS# 1

03/10/2018 07:39:12 PM

QTY	ITEM	TOTAL
1	Delivery	0.00
1	10 McNuggets EVM	10.29
1	Barbeque Sauce	
1	Barbeque Sauce	
1	L Coke	0.40
1	M French Fries	
1	HM GrChs S-Fry Yog	3.99
1	Snoopy	
1	HM Apple Juice	
1	6 Cookies	2.89
6	Choc Chip Cookie	
1	Baked Apple Pie	1.19
1	L French Fries	3.29
1	L Iced Tea	2.99
	Subtotal	25.04
	GST	1.11
	Take-Out Total	26.15
	CREDIT SALES - McDelivery	26.15
	Change	0.00


GST #: 120907092

SALE #04kdfwg532

FOR THIS OVER! PLEASE TURN THIS OVER! PLEASE TURN THIS OVER! PLEASE TURN THIS OVER! PLEASE TURN THIS OVER! PLEASE TURN THIS OVER!


可以看到，答案中不仅显示出了对应的图片，也给出了正确的答案，这也要归功于 OpenAI 对于任意文本强大的处理能力。

我们可以单独解析一下图片，看看对应的文本内容是什么。

 复制代码

```
1 output_image = image_parser.parse_file('./data/receipts/1100-receipt.jpg')
2 print(output_image.text)
```

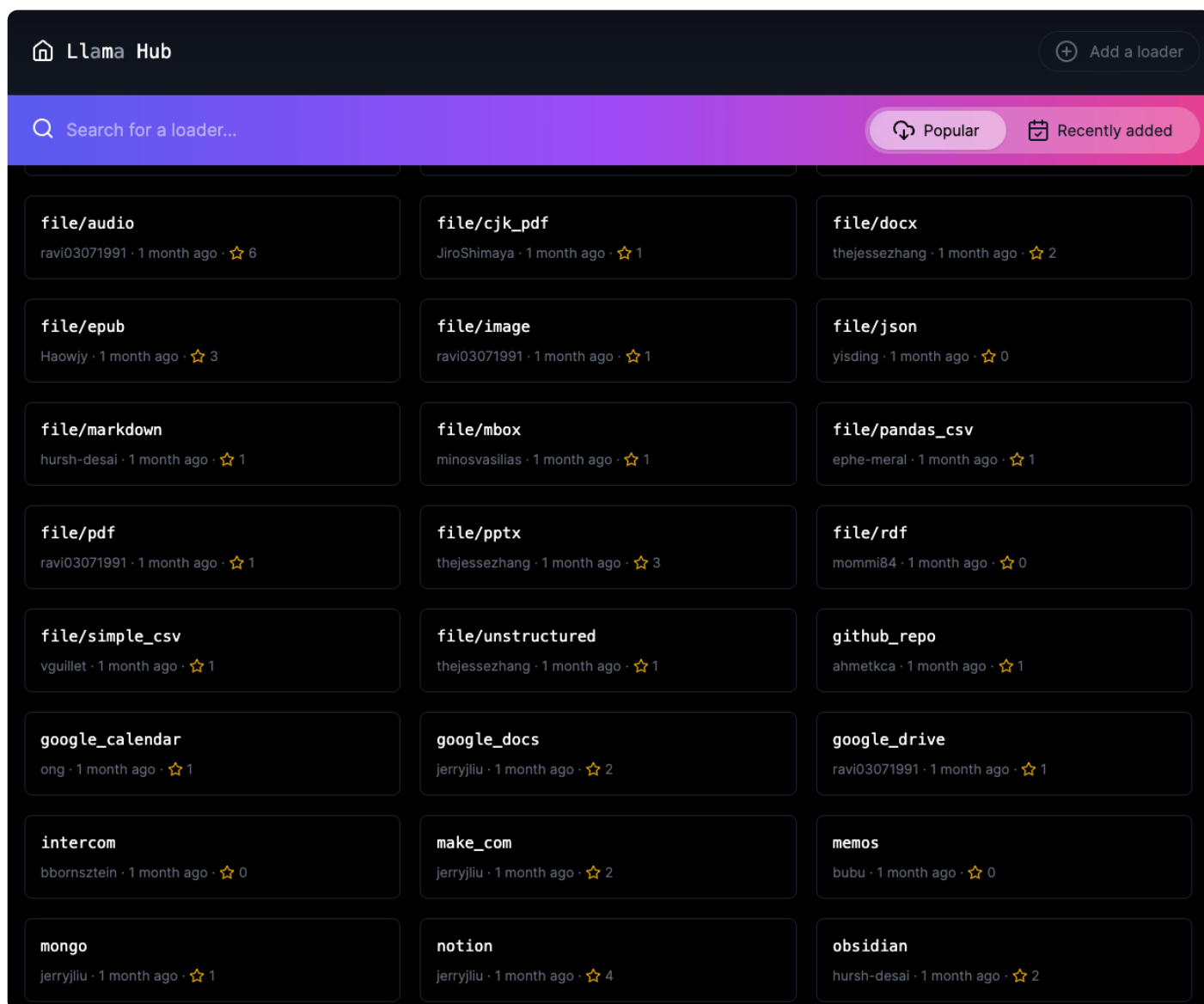
输出结果：

 复制代码

```
1 <s_menu><s_nm> Story</s_nm><s_num> 16725 Stony Platin Rd</s_nm><s_num> Store#:</s
2
3
4
```

可以看到，对应的就是 OCR 后的文本结果，里面的确有对应我们去的店铺的名字和时间，以及消费的金额。

围绕 OpenAI 以及整个大语言模型的生态还在快速发展中，所以 llama-index 这个库也在快速迭代。我自己在使用的过程中，也遇到各种各样的小 Bug。对于中文的支持也有各种各样的小缺陷。不过，作为开源项目，它已经有一个很不错的生态了，特别是提供了大量的 DataConnector，既包括 PDF、ePub 这样的电子书格式，也包括 YouTube、Notion、MongoDB 这样外部的数据源、API 接入的数据，或者是本地数据库的数据。你可以在 [llamahub.ai](https://llamahub.ai) 看到社区开发出来的读取各种不同数据源格式的 DataConnector。



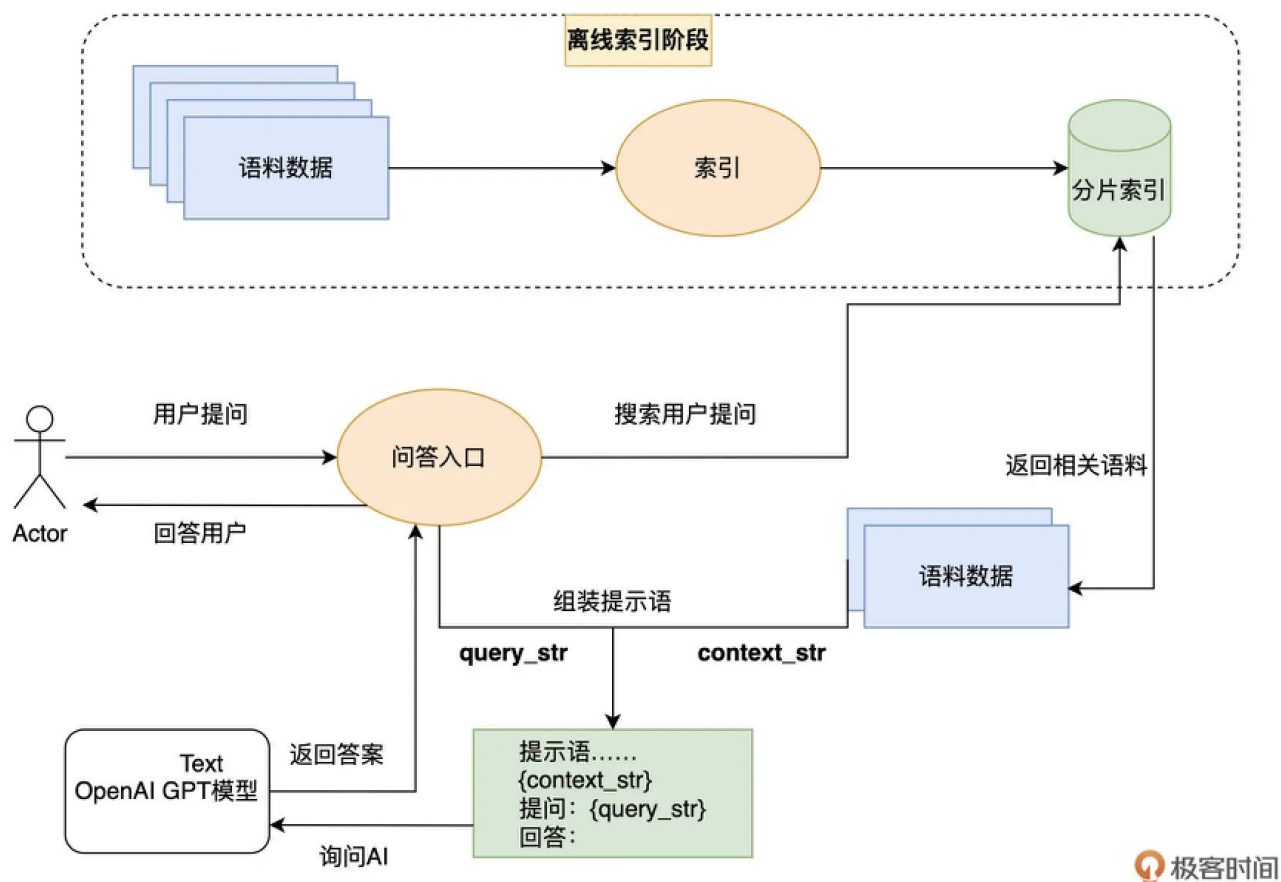
## 小结

好了，相信经过这一讲，你已经能够上手使用 llama-index 这个 Python 包了。通过它，你可以快速将外部的资料库变成索引，并且通过它提供的 query 接口快速向文档提问，也能够通过将文本分片，并通过树状的方式管理索引并进行小结。

llama-index 还有很多其他功能，这个 Python 库仍然在发展过程中，不过已经非常值得拿来使用，加速你开发大语言模型类的相关应用了。相关的文档，可以在 [官网](#) 看到。对应的代码也是开源的，遇到问题也可以直接去 [源代码](#) 里一探究竟。

llama-index 其实给出了一种使用大语言模型的设计模式，我称之为“第二大脑”模式。通过先将外部的资料库索引，然后每次提问的时候，先从资料库里通过搜索找到有相关性的材料，

然后再通过 AI 的语义理解能力让 AI 基于搜索到的结果来回答问题。



极客时间

其中，前两步的索引和搜索，我们可以使用 OpenAI 的 Embedding 接口，也可以使用其它的大语言模型的 Embedding，或者传统的文本搜索技术。只有最后一步的问答，往往才必须使用 OpenAI 的接口。我们不仅可以索引文本信息，也可以通过其他的模型来把图片变成文本进行索引，实现所谓的多模态功能。

希望通过今天的这几个例子，你也能开始建立起自己的“第二大脑”资料库，能够将自己的数据集交给 AI 进行索引，获得一个专属于你自己的 AI。

## 课后练习

1. llama-index 的生态，不仅支持各种各样的 DataConnector 去加载数据，后端也支持各种形式的索引，比如在语义搜索里面我们介绍过的 Faiss、Pinecone、Weaviate 它都是支持的。除了这些之外，你能看看 llama-index 还有哪些形式的索引吗？除了进行问答和文章概括之外，你觉得这个库还能帮助我们做什么事情？

2. 现在有很多应用，在你把文档上传之后，还会给你一系列的提示，告诉你可以向对应的书或者论文问什么问题。比如 [🔗 SCISPACE](#)，你能想想这些问题是怎么来的吗？

期待能在评论区看到你的分享，也欢迎你把这节课分享给感兴趣的朋友，我们下一讲再见。

## 推荐阅读

llama-index 的功能非常强大，并且源代码里也专门提供了示例部分。你可以去看一下它的官方文档以及示例，了解它可以用来干什么。

1. 官方文档: [🔗 https://gpt-index.readthedocs.io/en/latest/](https://gpt-index.readthedocs.io/en/latest/)
2. 源码以及示例: [🔗 https://github.com/jerryliu/llama\\_index](https://github.com/jerryliu/llama_index)

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

## 精选留言 (48)



**Wise**

2023-05-07 来自黑龙江

在llama\_index V0.6.1 版本中，没有GPTSimpleVectorIndex 类了

```
import openai, os
```

```
from llama_index import GPTVectorStoreIndex, SimpleDirectoryReader
```

```
os.environ["OPENAI_API_KEY"] = "
```

```
# 加载 documents
```

```
documents = SimpleDirectoryReader('./data/mr_fujino').load_data()
```

```
index = GPTVectorStoreIndex.from_documents(documents)
```

```
index.storage_context.persist('index_mr_fujino')
```

```
# 从磁盘重新加载:
```

```
from llama_index import StorageContext, load_index_from_storage
```

```
# rebuild storage context
```

```
storage_context = StorageContext.from_defaults(persist_dir='./index_mr_fujino')
```

```
# load index
```

```
index = load_index_from_storage(storage_context)
```

```
query_engine = index.as_query_engine()
response = query_engine.query("鲁迅先生在日本学习医学的老师是谁? ")
print(response)
```

参考官方文档连接: [https://gpt-index.readthedocs.io/en/latest/getting\\_started/starter\\_example.html](https://gpt-index.readthedocs.io/en/latest/getting_started/starter_example.html)

作者回复: 👍

如果还想要用0.5.x运行, 参看以下

"llama index 最近又更新了大版本, 接口又改了一遍。如果要立刻可以运行, 可以先 `pip install --force-reinstall -v "llama-index==0.5.27"` 退回到 0.5 系列的版本"

晚点我看一下更新代码到0.6.x 版本

共 4 条评论 >

👍 12



**hello**

2023-04-06 来自湖南

想请教下老师, 我们喂的语料, 会被其他人看到使用吗?

作者回复: 根据OpenAI的协议

1. 通过ChatGPT界面提交的会被内部看到, 审核, 并用于训练
2. 通过API提交的不会用于训练

共 3 条评论 >

👍 7



**Terry**

2023-04-25 来自浙江

老师, 请教一下langchain我理解也是一个LLM应用框架, 它的功能和版本也更新很快。它和llama\_index的区分是什么? 在LLM应用开发上, 我们一般怎么选择会比较好?

作者回复: llama\_index的重点放在了Index上, 也就是通过各种方式为文本建立索引, 有通过LLM的, 也有很多并非和LLM相关的。

langchain的重点在 agent 和 chain 上, 也就是流程组合上。



可以根据你的应用组合两个，如果你觉得问答效果不好，可以多研究一下llama-index。如果你希望有更多外部工具或者复杂流程可以用，可以多研究一下langchain。



👍 4



**daz2yy**

2023-04-04 来自广东

老师，请问下一个问题，我把它用作 QA 系统的时候有个问题，原本 QA 就有标准的回答模版，里面包括有文档地址、操作步骤等；如果让 GPT 根据这个模版来回答问题，他会自由发挥，会漏掉一部分内容；想拥有 AI 自由对话的能力，又想有固定的回答模版这个怎么能较好的兼顾呢？

作者回复：可以通过Few-Shot的方式，在Prompt里面给AI一些例子，类似于下面这样，具体Prompt你自己调了多试一下。

====

我们的问题一般用这样的格式回答：

问题：blablabla

回答：

1. 文档地址 blabla
2. 操作步骤 blabla

===

以下是上下文

{context\_str}

问题：{question\_str}

回答：



👍 4



**马听**

2023-04-12 来自上海

分享一个加载MySQL数据的例子：

```
from llama_index import GPTSimpleVectorIndex,download_loader
```

```
DatabaseReader = download_loader('DatabaseReader')
```

```
reader = DatabaseReader(  
    scheme = "mysql", # Database Scheme  
    host = "localhost", # Database Host  
    port = "3306", # Database Port  
    user = "martin", # Database User  
    password = "xxxxxx", # Database Password  
    dbname = "martin", # Database Name  
)
```

```
query = f"""  
select * from student_info  
"""
```

```
documents = reader.load_data(query=query)  
print(documents)
```

作者回复: 👍

共 3 条评论 >

👍 3



**Oxygen Au 听**

2023-05-09 来自美国

```
response = list_index.query("下面鲁迅先生以第一人称'我'写的内容，请你用中文总结一下:",  
response_mode="tree_summarize")  
print(response)
```

上面这段代码报错，AttributeError: 'GPTListIndex' object has no attribute 'query'，我用的是llama-index 0.6.1

下面是正确的代码

```
query_engine = list_index.as_query_engine(response_mode="tree_summarize")  
response = query_engine.query("下面鲁迅先生以第一人称'我'写的内容，请你用中文总结一下:")  
print(response)
```

结果：鲁迅先生在日本学习医学时遇到了藤野严九郎教授，他很有学问，对学生也很关心，甚至帮助鲁迅修改讲义。但鲁迅当时不够用功，有时也很任性。在学习中，他遇到了一些困难和不愉快的事情，最终决定离开医学去学习生物学。离开前，藤野先生送给他一张照片，并希望他能保持联系。鲁迅很久没有和任何人通信，但想起了这位热心的老师，他的照片挂在鲁迅的房间里，每当他疲倦时看到照片就会感到勇气和良心发现。

作者回复: llama index 最近又更新了大版本，接口又改了一遍。如果要立刻可以运行，可以先 `pip install --force-reinstall -v "llama-index==0.5.27"` 退回到 0.5 系列的版本

晚点我看一下更新代码到0.6.x 版本



👍 2



**Viola**

2023-04-05 来自摩尔多瓦

有同学遇到吗？

type object 'GPTSimpleVectorIndex' has no attribute 'from\_documents'

作者回复: 更新一下 llama-index 的版本，llama-index的接口从 0.4 到 0.5 做了比较大的更新。这一讲的内容，也是根据 0.5 的更新重新改过的。用最新版吧。

`pip install -U llama-index`

共 2 条评论 >

👍 2



**hawk**

2023-04-04 来自福建

这些预选的问题，应该也是通过组合特定的提示语，和段落摘要，扔给GPT得到的吧？

作者回复: 是的，组合就是通过 PromptTemplate 实现的

共 3 条评论 >

👍 2



**黄智荣**

2023-05-16 来自福建

现在有很多应用，在你把文档上传之后，还会给你一系列的提示，告诉你可以向对应的书或者论文问什么问题。

----- 可以根据索引的文本，让chatgpt 设计几个提问的问题

作者回复: 👍，这个一般被称之为 Self-Ask，的确是一个好办法。



👍 1



**东临沧海**

2023-05-11 来自浙江

建议老师安装的库文件标明一下版本号，遇到这样问题好几次了，每次都要浪费大量时间。  
llama\_index版本都到v0.6.5，更新太快了

作者回复: 这周会更新一下github里的pip requirements文件，锁住版本号。  
但是想llama-index这样不讲武德每次小版本更新都不向前兼容的函数库的确也不常见.....

再贴一次吧:

llama index 最近又更新了大版本，接口又改了一遍。如果要立刻可以运行，可以先 `pip install --force-reinstall -v "llama-index==0.5.27"` 退回到 0.5 系列的版本

晚点我看一下更新代码到0.6.x 版本



👍 1



**snow**

2023-05-09 来自日本

想请教下老师，为什么我在调用query 的时候，提示说BaseQueryEngine.query() got an unexpected keyword argument 'response\_mode'？发现当定义 `list_index = GPTListIndex(nodes=nodes, service_context=service_context)`，没法直接调用`list_index.query()`，而需要 `list_index.as_query_engine().query()` 这样子调用。

作者回复: llama index 最近又更新了大版本，接口又改了一遍。如果要立刻可以运行，可以先 `pip install --force-reinstall -v "llama-index==0.5.27"` 退回到 0.5 系列的版本

晚点我看一下更新代码到0.6.x 版本

共 2 条评论 >

👍 1



**勇.Max**

2023-05-08 来自北京

上面from llama\_index import GPTSimpleVectorIndex会报错，因为现在已经改成了GPTVectorStoreIndex。

from llama\_index import GPTVectorStoreIndex //老师看到后可以更新下

编辑回复: 收到，马上更新，感谢提醒🍷

共 3 条评论 >

👍 1



**橘子汽水**

2023-05-03 来自广东

重新输入了

!pip install -U llama-index

还是报错，老师可以帮忙看看吗？

ImportError: cannot import name 'GPTSimpleVectorIndex' from 'llama\_index' (/usr/local/lib/python3.10/dist-packages/llama\_index/\_\_init\_\_.py)

作者回复: llama index 最近又更新了大版本，接口又改了一遍。如果要立刻可以运行，可以先 pip install --force-reinstall -v "llama-index==0.5.27" 退回到 0.5 系列的版本

晚点我看一下更新代码到0.6.x 版本

共 4 条评论 >

👍 1



**Dev.lu**

2023-04-13 来自新加坡

谢谢老师分享，想问一下，

1. 语料的大小会不会影响回答问题的延时？
2. 如果需要对语料进行添加或者减少，每次都需要重跑代码？
3. 如果离线语料数据很大，index过后的语料数据有没有可能导致内存不够，或者OpenAI的API会限制request的大小吗？还是收费不同

作者回复: 1. 不影响网络

2. 不需要，但是需要维护好一个 embedding vector的索引。可以用向量数据库，也可以自己在内存里面管理

3. 一般不至于，你可以算一下，一条数据一般也就3-4K，1百万条数据才3G数据。但是需要比较多时

间建立索引，这个和OpenAI的request大小无关，一般即使用batch一个batch也就拿1千条数据的Embedding吧



**xbc**

2023-04-04 来自海南

老师，带我们玩玩 modelscope.cn

作者回复: modelscope.cn 基本上就是学习 huggingface.co 呀



**狸猫酱**

2023-05-31 来自广东

想问一下老师，这种方式是不是只适合数据量小的资料库？有没有一个适用的大小范围呢？另外比如我想做某个专有领域的对话机器人，比如化学研究，那我把所有能找到的化学相关的论文、教材等相关资料和gpt结合，Llama Index的方式是否还适用呢？



**农民园丁**

2023-05-27 来自立陶宛

请问老师两个问题：1.如果本地知识库是英文的，能用中文检索到吗？2. 如果我有几本书建立了本地知识库，但是内容有重复的，这时候会得出什么样的结果？



**Geek\_378f83**

2023-05-22 来自美国

请教：运行代码时，报错，搜索和问GPT也没解决：

ImportError

Traceback (most recent call last)

<ipython-input-16-aefa1f0d78bc> in <cell line: 2>()

1 import openai, os

----> 2 from llama\_index import GPTSimpleVectorIndex, SimpleDirectoryReader, ServiceContext

3

4 openai.api\_key = os.environ.get("OPENAI\_API\_KEY")

5



ImportError: cannot import name 'GPTSimpleVectorIndex' from 'llama\_index' (/usr/local/lib/python3.10/dist-packages/llama\_index/\_\_init\_\_.py)

NOTE: If your import is failing due to a missing package, you can manually install dependencies using either !pip or !apt.

To view examples of installing some common dependencies, click the "Open Examples" button below.



**小耿**

2023-05-22 来自北京

请问老师，我换了一篇文章用 llama\_index 做总结，一直报限速问题，好像是llama内部调用的报错，应该怎么改进代码呢？

Retrying langchain.chat\_models.openai.ChatOpenAI.completion\_with\_retry.<locals>.\_completion\_with\_retry in 8.0 seconds as it raised RateLimitError: Rate limit reached for default-gpt-3.5-turbo in organization org-oEUgEoq9uZ4oju2xVxmZ7cUO on requests per min. Limit: 3 / min. Please try again in 20s. Contact us through our help center at help.openai.com if you continue to have issues. Please add a payment method to your account to increase your rate limit. Visit <https://platform.openai.com/account/billing> to add a payment method..

作者回复: 你应该用的是免费API的额度，现在限速是一分钟调用3次API。看看是否周围有朋友有付费API的账户拿来用一下吧。



**牛味浓龙魏流**

2023-05-15 来自四川

所以llama\_index跟chatGPT没什么关系是吗。。还是说这个包本身也在跟openAI打交道，看代码看不出来

作者回复: 没什么关系，只是封装了一部分对于OpenAI的调用，你也可以不调用OpenAI的API，调用其他的大语言模型，比如 Claude 或者 Cohere。



