

## ChatGPT如何通过ChatGPT问一本书的问题？

周末程序猿    
腾讯科技（深圳）有限公司 工程师

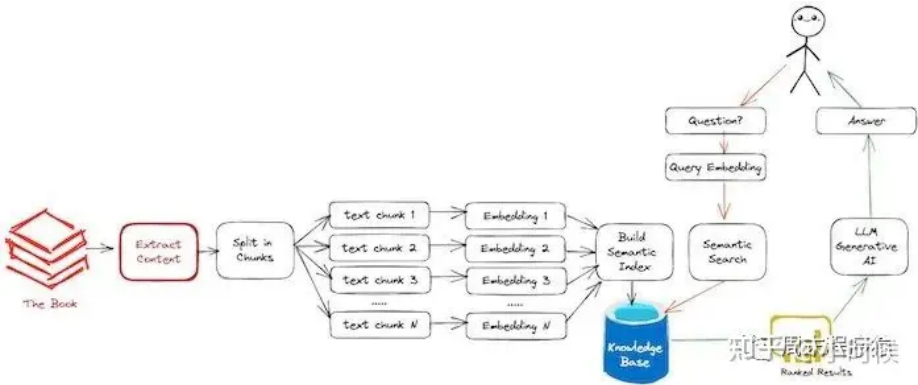
2 人赞同了该文章

在很多场景下需要私域数据，但是在使用ChatGPT对话回答是很泛或者没有相关答案，因此你就需要自己喂养数据，然后形成自己的私域数据数据集，以下就是用一本书作为例子，通过输入一本书问ChatGPT关于这本书其中的问题。其步骤如下：

- (1) 提取书中的内容；
- (2) 将书分为小块；
- (3) 建立语义索引；
- (4) 问书中的问题；

本文使用的是开源的书：pure-bash-bible中译版([github.com/dylanaraps/p...](https://github.com/dylanaraps/pure-bash-bible))

### 提取书中的内容



需要使用上一篇文章的LangChain库中和document\_loaders模块。其中text\_splitter库包含允许用户访问和操作来自不同来源的文本数据的函数和类。

- a. UnstructuredPDFLoader，它用于从PDF文件加载和提取文本，PDF文件的路径指定为“xxxx.pdf”；
- b. UnstructuredMarkdownLoader，它用于从Markdown文件加载和提取文本；
- ... 还有一些其他加载非结构化数据的库使用。

代码如下：

```
loader = UnstructuredMarkdownLoader("../docs/books/pure-bash-bible.md")
data = loader.load() # 加载文件数据
print(f'You have {len(data)} document(s) in your data')
print(f'There are {len(data[0].page_content)} characters in your document')
```

通过以上代码可以加载您需要训练的markdown文件，打印文本中大概多少字符。


### 将书分为小块

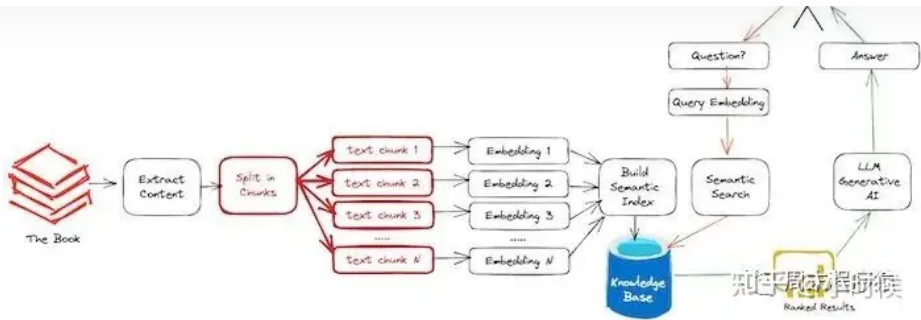
×

登录即可查看 **超5亿** 专业优质内容

超 5 千万创作者的优质提问、专业回答、深度文章和精彩视频尽在知乎。

立即登录/注册





我们将把加载的文档分成更小的“页面”，每个页面有500个字符，目的是在我们向OpenAI提出问题时向它提供上下文信息。与其让 OpenAI 在我们每次提出问题时就阅读整本书，不如给它一小段相关信息来处理，这样效率更高、成本效益更高。

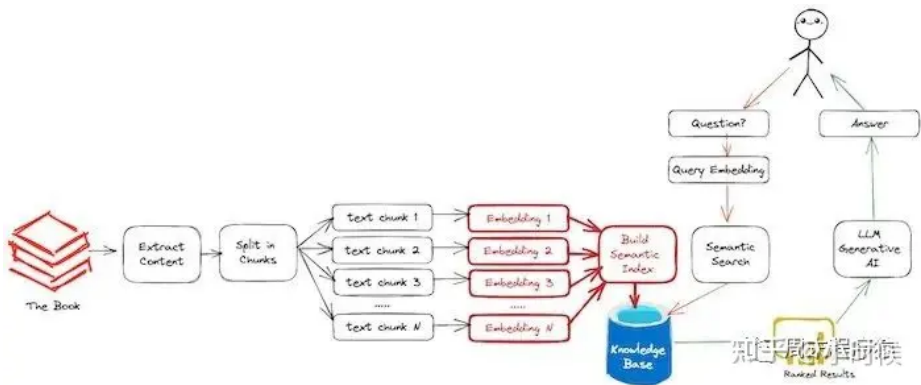
代码如下：

```
text_splitter = RecursiveCharacterTextSplitter(
    chunk_size=500, chunk_overlap=200)
docs = text_splitter.split_documents(data)

print(f'Now you have {len(docs)} documents')
```

现在我们已经完成了文本的准备工作，可以开始下一步了。

建立语义索引



创建文档的嵌入，为语义搜索做好准备，将使用向量库，这样就可以将更多书籍添加到我们的语料库中，而不必每次都重新阅读原始文档，这里我们使用向量库FAISS。既然说到向量库，那么先展开说一说什么是向量库。

**向量库：**在现实生活中非结构化数据的体量变得越来越多，所谓结构化数据指的是像音频、视频、图片，包括用户信息、分子式、时序数据，地理位置数据等，这些数据相比传统的结构化数据更贴合人的需求，它的描述能力更强，同时处理难度也更高。根据IDC的调查显示，未来80%以上的数据可能都是非结构化数据。但目前非结构化数据并没有被很好地利用，以往都是通过AI去处理，各种各样的非结构化数据通过深度学习模型转化为一组高维的稠密的数据，这组数据通过他们最近邻的关系，就可以更好地去匹配。

**FAISS库：**是由Facebook开发的适用于稠密向量匹配的开源库，支持点积、欧氏距离等，同时支持精确检索与模糊搜索。FAISS的使用是围绕index中包含了被索引的数据库向量以及对应的索引值。在构建index时，向量的维度d，随后通过add()的方式将被检索向量存入index中，最终

×

登录即可查看 超5亿 专业优质内容

超 5 千万创作者的优质提问、专业回答、深度文章和精彩视频尽在知乎。

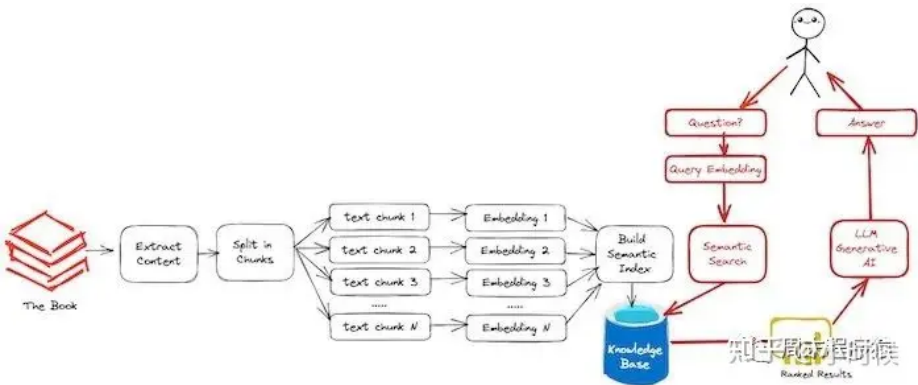
知乎

```
embeddings = OpenAIEmbeddings(openai_api_key=OPENAI_API_KEY)
store = FAISS.from_documents(docs, embeddings)
faiss.write_index(store.index, "langchain-pure-base-bible.index")
store.index = None

with open("faiss-book.pkl", "wb") as f:
    pickle.dump(store, f)
```

至此，我们已经将这本书分解成更小的文档，使用OpenAI嵌入将每个文档转换为一个向量，我们可以继续下一步，即构建实际的问答组件。

问书中的问题



建立索引后，我们就可以查询这些文档以获取我们的答案。代码如下：

```
// 先加载前面训练好的语义索引
index = faiss.read_index(index_name)
with open(namespace, "rb") as f:
    docsearch = pickle.load(f)
docsearch.index = index

// 使用ChatOpenAI接口
llm = ChatOpenAI(
    temperature=0, openai_api_key=OPENAI_API_KEY, verbose=True)
chain = load_qa_chain(llm, chain_type="stuff")

// 调用OpenAI根据前面切割好的文档上下文回答问题
query_list = ["可以给一下shell的示例用法的代码么?", "for循环该如何使用?", "如何检查一个
for query in query_list:
    print("问题: ", query)
    docs = docsearch.similarity_search(query, include_metadata=True)
    r = chain.run(input_documents=docs, question=query)
    print("结果: ", r.encode('utf-8').decode('utf-8'))
```

ChatGPT回答如下：

问题：本书中条件表达式有哪些？

结果：本书中介绍了以下条件表达式：

- if-then
- if-then-else
- if-elif-else
- case-in-esac
- 三元条件表达式

×

登录即可查看 超5亿 专业优质内容

超 5 千万创作者的优质提问、专业回答、深度文章和精彩视频尽在知乎。

从上可以看出ChatGPT并不是直接用原文来回答，是的将文档模型建立好，总结提炼出的回答。

到目前为止，如何问一本书的问题步骤介绍完成，我们也可以采取许多其他步骤来改进系统并使其更加有效。例如，我们可以通过在更大的问题和答案数据集上进行训练来提高系统的准确性，我们还可以在特定领域（例如，历史、科学等）上微调OpenAI模型，以提高其在该领域相关的问题上的表现。（可以结合LangChain的介绍来做更多有趣的事情）

## 资料

(1) [bennychung.github.io/a...](https://bennychung.github.io/a...)

(2) 代码如下：

```
from langchain.chains.question_answering import load_qa_chain, load_summarize_chain
from langchain.chat_models import ChatOpenAI # 使用高版本的langchain
from langchain.embeddings.openai import OpenAIEmbeddings
from langchain.vectorstores import FAISS
from langchain.document_loaders import UnstructuredMarkdownLoader
from langchain.text_splitter import RecursiveCharacterTextSplitter
import os
import faiss
import pickle

OPENAI_API_KEY = os.environ["API_SECRET"]

index_name = "langchain-pure-base-bible.index"
namespace = "faiss-book.pkl"

def isExistTrainFile():
    return os.path.exists(index_name)

def train():
    loader = UnstructuredMarkdownLoader("../docs/books/pure-base-bible.md")
    data = loader.load()
    print(f'You have {len(data)} document(s) in your data')
    print(f'There are {len(data[0].page_content)} characters in your document')

    text_splitter = RecursiveCharacterTextSplitter(
        chunk_size=1000, chunk_overlap=0)
    docs = text_splitter.split_documents(data)

    print(f'Now you have {len(docs)} documents')

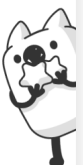
    embeddings = OpenAIEmbeddings(openai_api_key=OPENAI_API_KEY)
    store = FAISS.from_documents(docs, embeddings)
    faiss.write_index(store.index, index_name)
    store.index = None

    with open(namespace, "wb") as f:
        pickle.dump(store, f)

def runPrompt():
    index = faiss.read_index(index_name)
    with open(namespace, "rb") as f:
        docsearch = pickle.load(f)
    docsearch.index = index
```

登录即可查看 **超5亿** 专业优质内容

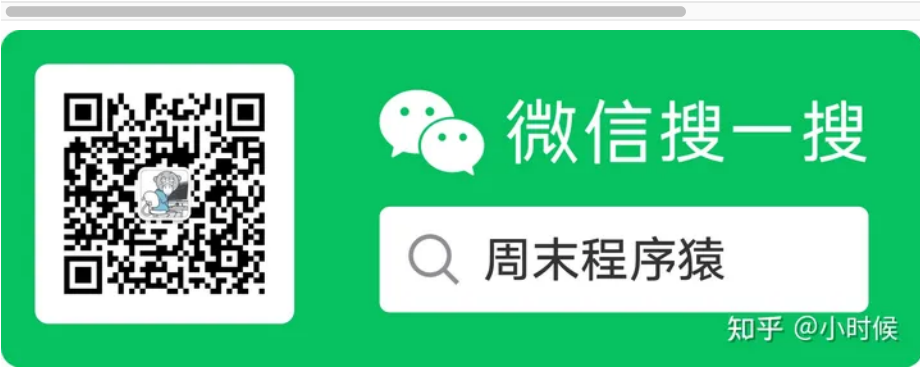
超 5 千万创作者的优质提问、专业回答、深度文章和精彩视频尽在知乎。



知乎

```
query_list = ["可以给一下shell的示例用法的代码么?", "for循环该如何使用?", "如何检
for query in query_list:
    print("问题: ", query)
    docs = docsearch.similarity_search(query, include_metadata=True)
    r = chain.run(input_documents=docs, question=query)
    print("结果: ", r.encode('utf-8').decode('utf-8'))

if __name__ == "__main__":
    if not isExistTrainFile():
        train()
    runPrompt()
```



发布于 2023-04-04 12:46 · IP 属地广东

[OpenAI](#)

写下你的评论...



还没有评论，发表第一个评论吧

推荐阅读

【ICF教练知识库】ORID焦点提问法 常用的20个问题

如何找到真问题 问题how/why/

登录即可查看 **超5亿** 专业优质内容

超 5 千万创作者的优质提问、专业回答、深度文章和精彩视频尽在知乎。



知乎

登录即可查看 **超5亿** 专业优质内容

超 5 千万创作者的优质提问、专业回答、  
深度文章和精彩视频尽在知乎。

