# 检索器的定义和使用

检索器是什么？它与向量存储有什么区别？本文尝试解释检索器的定义以及如何使用。

#### 检索器的定义

检索器是一种可以通过非结构化查询返回文档的接口。它比向量存储的应用范围更广。一个检索器并不需要能够存储文档，只需要能够返回（或检索）文档即可。向量存储可以作为检索器的支撑，但是也存在其他类型的检索器。

#### 检索器的使用

LangChain中的BaseRetriever类的公开API非常简单。

from abc import ABC, abstractmethod  
from typing import Any, List  
from langchain.schema import Document  
from langchain.callbacks.manager import Callbacks  
  
class BaseRetriever(ABC):  
 ...  
 def get\_relevant\_documents(  
 self, query: str, \*, callbacks: Callbacks = None, \*\*kwargs: Any  
 ) -> List[Document]:  
 """Retrieve documents relevant to a query.  
 Args:  
 query: string to find relevant documents for  
 callbacks: Callback manager or list of callbacks  
 Returns:  
 List of relevant documents  
 """  
 ...  
  
 async def aget\_relevant\_documents(  
 self, query: str, \*, callbacks: Callbacks = None, \*\*kwargs: Any  
 ) -> List[Document]:  
 """Asynchronously get documents relevant to a query.  
 Args:  
 query: string to find relevant documents for  
 callbacks: Callback manager or list of callbacks  
 Returns:  
 List of relevant documents  
 """  
 ...

你可以调用get\_relevant\_documents或者异步的get\_relevant\_documents方法来检索与查询相关的文档，其中”relevance”是由你正在调用的特定检索器对象定义的。

我们主要关注的检索器类型是向量存储检索器，我们将重点介绍这个类型。

为了理解什么是向量存储检索器，理解向量存储的概念是非常重要的。所以我们首先来看看向量存储是什么。

默认情况下，LangChain使用Chroma作为向量存储来索引和搜索嵌入。在本文中，我们首先需要安装chromadb。

pip install chromadb

我们选择了在文档上进行问题回答的示例来进行演示，因为它很好地整合了许多不同的元素（文本分割器，嵌入，向量存储），并展示了如何将它们串联使用。

文档上的问题回答包括四个步骤：

1. 创建索引
2. 从索引创建检索器
3. 创建问题回答链
4. 提出问题！

每一个步骤都有多个子步骤和潜在的配置。在本指南中，我们将主要关注步骤（1）。我们将首先展示一个一行代码的示例，然后详细解析其中的内容。

首先，让我们导入公共类。

from langchain.chains import RetrievalQA  
from langchain.llms import OpenAI

在通用设置中，接下来我们来指定我们想要使用的文档加载器。

from langchain.document\_loaders import TextLoader  
loader = TextLoader('../state\_of\_the\_union.txt', encoding='utf8')

为了尽快开始，我们可以使用VectorstoreIndexCreator。

from langchain.indexes import VectorstoreIndexCreator

index = VectorstoreIndexCreator().from\_loaders([loader])

Running Chroma using direct local API.  
 Using DuckDB in-memory for database. Data will be transient.

创建索引后，我们就可以使用它来提出数据问题了！注意，这里面实际上也做了几个步骤，我们将在后面的指南中介绍。

query = "What did the president say about Ketanji Brown Jackson"  
index.query(query)

" The president said that Ketanji Brown Jackson is one of the nation's top legal minds, a former top litigator in private practice, a former federal public defender, and from a family of public school educators and police officers. He also said that she is a consensus builder and has received a broad range of support from the Fraternal Order of Police to former judges appointed by Democrats and Republicans."

query = "What did the president say about Ketanji Brown Jackson"  
index.query\_with\_sources(query)

{'question': 'What did the president say about Ketanji Brown Jackson',  
 'answer': " The president said that he nominated Circuit Court of Appeals Judge Ketanji Brown Jackson, one of the nation's top legal minds, to continue Justice Breyer's legacy of excellence, and that she has received a broad range of support from the Fraternal Order of Police to former judges appointed by Democrats and Republicans.\n",  
 'sources': '../state\_of\_the\_union.txt'}

VectorstoreIndexCreator返回的是VectorStoreIndexWrapper，它提供了查询和带来源的查询功能。 如果我们只想直接访问向量存储，我们也可以这样做。

index.vectorstore

<langchain.vectorstores.chroma.Chroma at 0x119aa5940>

如果我们只想直接访问VectorstoreRetriever，我们也可以这样做。

index.vectorstore.as\_retriever()

VectorStoreRetriever(vectorstore=<langchain.vectorstores.chroma.Chroma object at 0x119aa5940>, search\_kwargs={})

接下来，我们要深入了解一下索引是如何创建的。VectorstoreIndexCreator中隐藏了很多魔法。它做了什么呢？

在文档加载后，主要进行了三个步骤：

1. 将文档分割成块
2. 为每个文档创建嵌入
3. 在向量存储中存储文档和嵌入

让我们在代码中一步步解析这个过程。

加载文档。

documents = loader.load()

分割文档。

from langchain.text\_splitter import CharacterTextSplitter  
text\_splitter = CharacterTextSplitter(chunk\_size=1000, chunk\_overlap=0)  
texts = text\_splitter.split\_documents(documents)

创建嵌入。

from langchain.embeddings import OpenAIEmbeddings  
embeddings = OpenAIEmbeddings()

创建向量存储

from langchain.vectorstores import Chroma  
db = Chroma.from\_documents(texts, embeddings)

Running Chroma using direct local API.  
 Using DuckDB in-memory for database. Data will be transient.

创建索引。

retriever = db.as\_retriever()

我们像以前一样，创建问答链。

qa = RetrievalQA.from\_chain\_type(llm=OpenAI(), chain\_type="stuff", retriever=retriever)

query = "What did the president say about Ketanji Brown Jackson"  
qa.run(query)

" The President said that Judge Ketanji Brown Jackson is one of the nation's top legal minds, a former top litigator in private practice, a former federal public defender, and from a family of public school educators and police officers. He said she is a consensus builder and has received a broad range of support from organizations such as the Fraternal Order of Police and former judges appointed by Democrats and Republicans."

VectorstoreIndexCreator不过是一种封装了各种逻辑的工具。它的配置是灵活的，可以设定使用的文本分割器、嵌入方式以及使用的向量存储。例如，你可以按照以下方式对其进行配置：

index\_creator = VectorstoreIndexCreator(  
 vectorstore\_cls=Chroma,  
 embedding=OpenAIEmbeddings(),  
 text\_splitter=CharacterTextSplitter(chunk\_size=1000, chunk\_overlap=0)  
)

我们希望能揭示出VectorstoreIndexCreator内部的运行机制。我们认为提供一种简单的索引创建方式是重要的，但同时，理解这个过程中底层的操作也同样关键。