**通过LlamaIndex添加引用源（attribution）**

**解决RAG不准确问题**

当前，LLM 的最大问题就是缺乏最新的知识和特定领域的知识。对于这一问题，主要有两种解决方法：微调和检索增强生成（RAG）。业内许多公司都认为相比微调，RAG 是更好的解决方法。归根究底是因为微调的成本更高，需要使用的数据也更多，因此主要适用于风格迁移（Style Transfer）场景。相比之下，RAG 方法使用向量数据库，从而将知识和数据注入到应用中，更适用于通用场景。

采用RAG方法就意味着使用向量数据库存储真理数据，这样可以确保应用返回正确的信息以及知识，而不是在缺乏数据时产生幻觉，捏造回答。不过，随着越来越多的文档、用例等信息被注入应用中，越来越多开发者意识到信息来源的重要性，它可以确保信息准确性，使得大模型的回答更加真实。

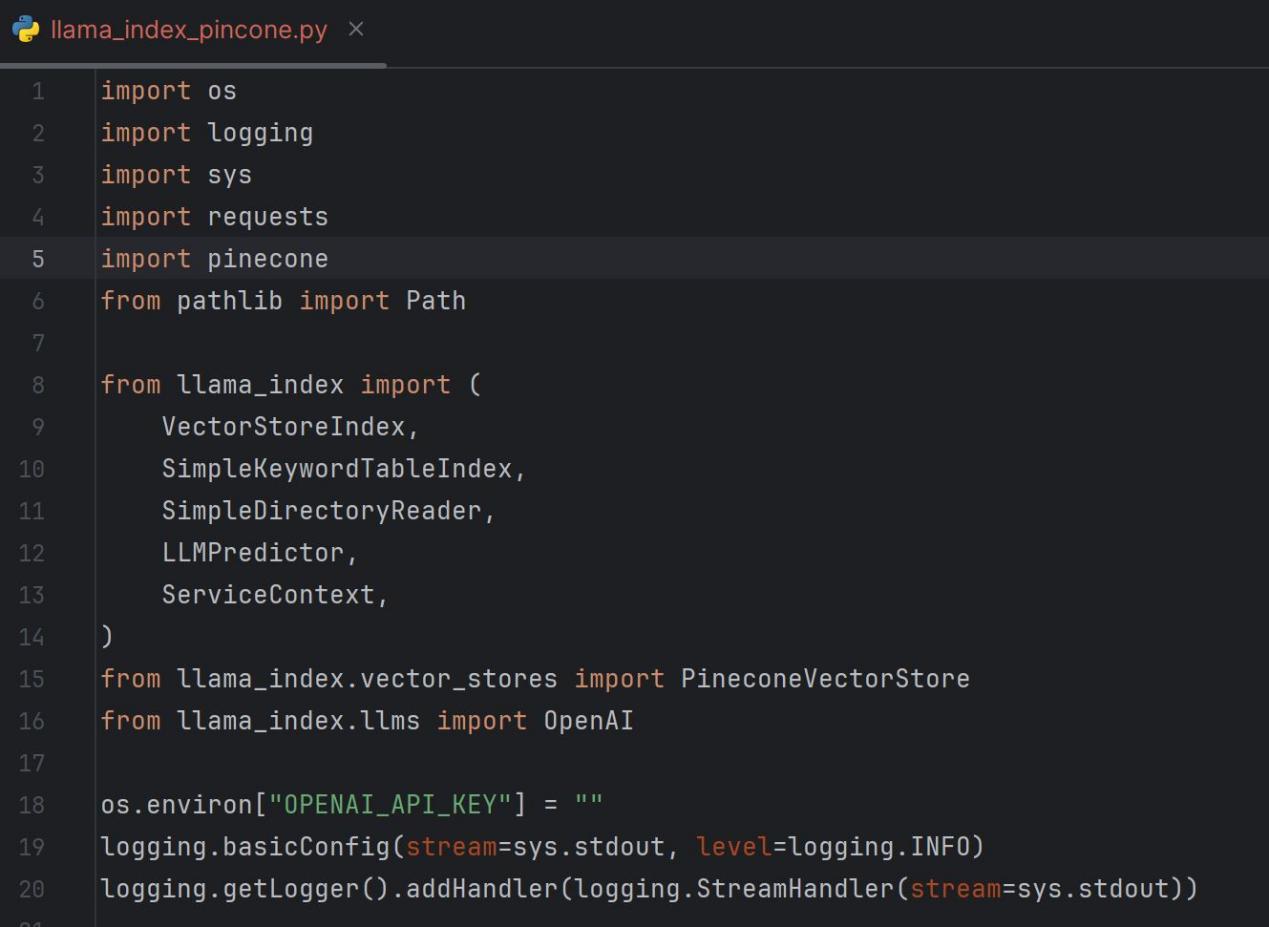
这就需要用到引用或者归属（attribution）。如果返回的响应带有引用或者归属，那么我们就可以了解该响应内容来自于哪个文档或文档中的哪个片段。因此，本文将详解为 LLM 加入引用的重要性，以及如何获取引用来源。

[RAG](https://link.zhihu.com/?target=https://zilliz.com/use-cases/llm-retrieval-augmented-generation" \t "https://zhuanlan.zhihu.com/p/_blank)引用源是一种通用解决方案，可为LLM应用的响应添加引用源，从而为响应提供更多上下文信息。那么如何为响应添加RAG引用源呢？其实有很多解决方法。既可以将文本块存储在向量数据库之中，也可以使用 [LlamaIndex](https://link.zhihu.com/?target=https://mp.weixin.qq.com/s?__biz=MzUzMDI5OTA5NQ==&mid=2247498133&idx=1&sn=d3c603265e969cbb64f4b2e1c288f22e&chksm=fa51582dcd26d13b3d9504677410886ab6de6f1719a63e150b8a7e2109b64dc265c61c0dc320&scene=21%23wechat_redirect" \t "https://zhuanlan.zhihu.com/p/_blank) 之类的框架。

**一、通过使用LlamaIndex和Pinecone为 LLM 响应添加引用源**

在本示例中，我们从百科中获取了不同城市的数据，并进行查询，最终获得带引用的响应。

首先，导入一些必要的库并加载 OpenAI API 密钥，同时也需要用到LlamaIndex的7个子模块。在本示例中，OpenAI用于访问LLM，CitationQueryEngine用于创建引用查询引擎，PineconeVectorStore用于将Pinecone作为向量存储数据库。此外，导入 VectorStoreIndex来使用Pinecone，SimpleDirectoryReader 用于读取本地数据，以及 StorageContext 和 ServiceContext用于访问Pinecone。最后，用load\_dotenv加载我们的OpenAI API密钥。



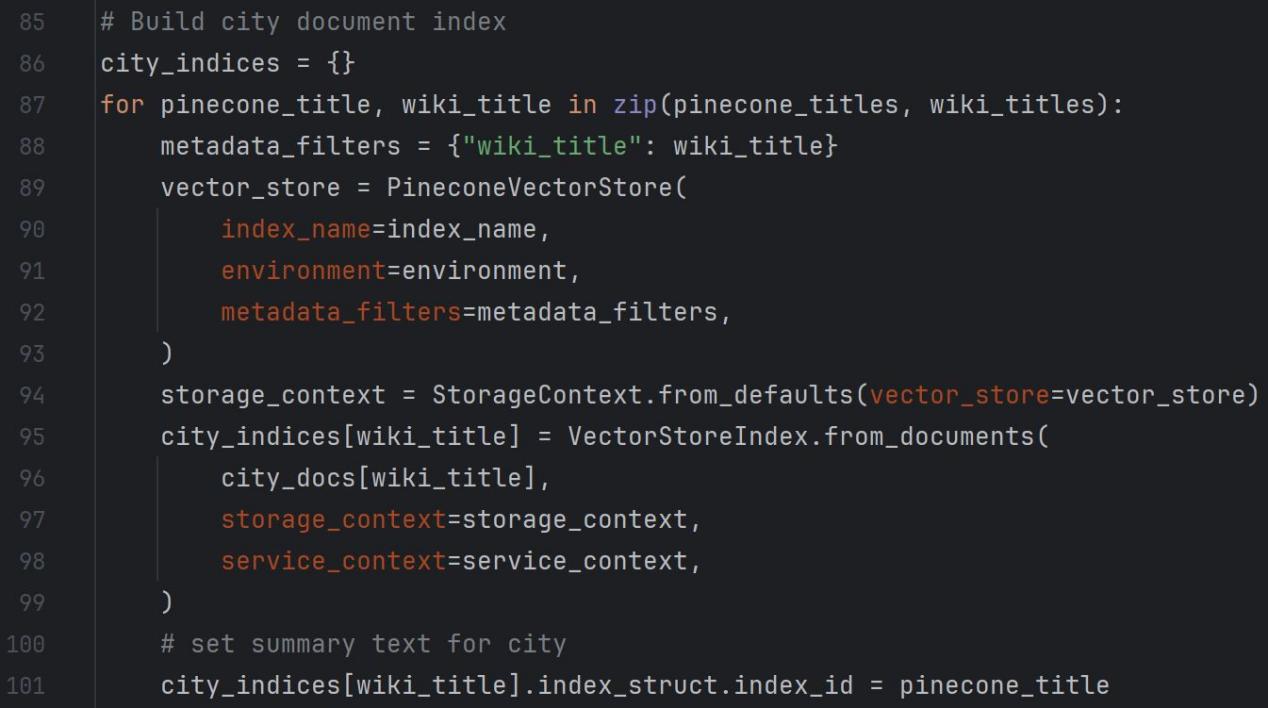
### **获取测试数据：**首先准备和处理数据，下面的代码从百科API中获取了wiki\_titles列表中**提到**的页面并将结果保存到本地文件中。



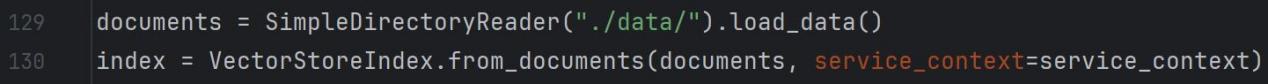
### 为索引创建上下文，从而帮助索引和检索器了解需要使用哪些服务。本例使用chatglm3-6b。此外，我们还需创建一个存储上下文，以便索引知道在哪里存储和查询数据。本例使用Pinecone进行向量存储。

### **捕获2**

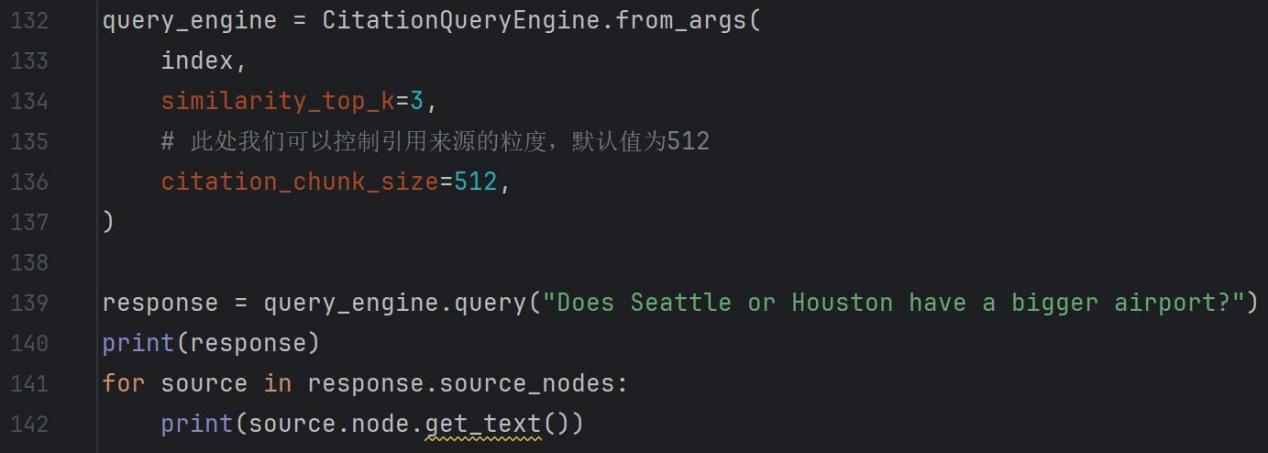
### **在 LlamaIndex 中设置向量存储（Vector Store）：**数据准备完成后，可以设置带应用逻辑。首先，我们需要启动向量数据库。在本例中，我们使用Pinecone，然后，用LlamaIndex的PineconeVectorStore模块连接Pinecone，将其作为向量存储。



### 设置完成后可以加载此前爬取的数据，并创建向量存储索引。

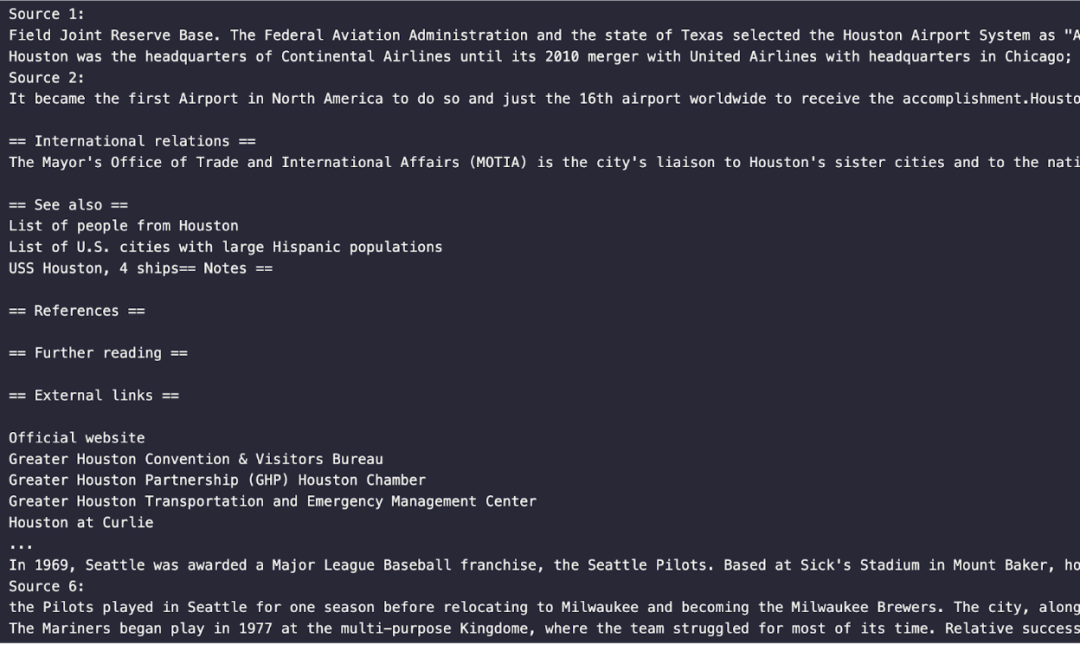


### **带引用的查询：**这一步可创建一个引用查询引擎。可以设置向量索引，返回结果数量以及引用文本块大小。随后运行查询命令。

****

## 具体来看，可使用LlamaIndex作为数据路由器，Pinecone作为向量存储来构建带有引用的RAG应用。本文提供的代码可先从百科上获取一些数据，然后启动一个Pinecone实例，然后在LlamaIndex中创建一个向量存储实例。将数据存入Pinecone中，然后使用LlamaIndex 构建引用查询引擎来追踪返回响应的归属和引用源。

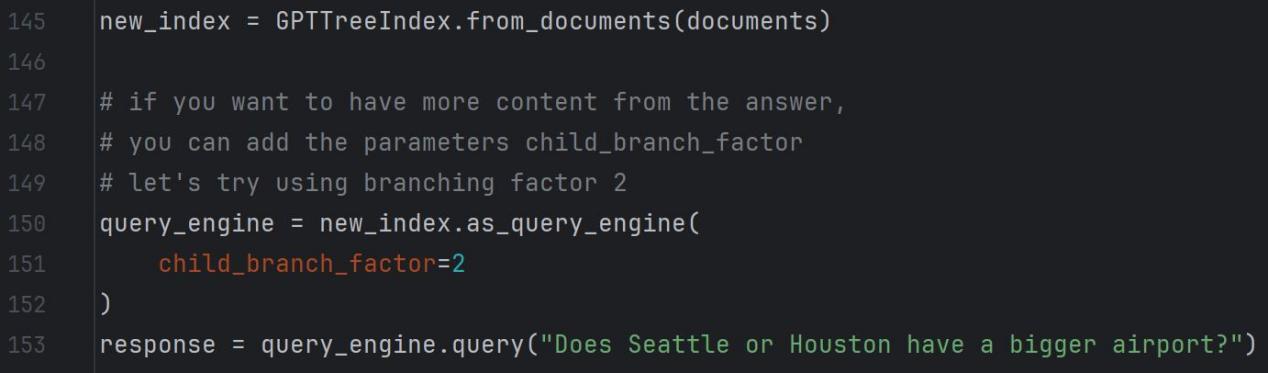
**响应如下所示：**

****

## **二、使用LlamaIndex构建索引**

### **1. 树索引（Tree Index）**

树索引是一个树结构索引，其中每个节点都是子节点的摘要。在索引构建过程中，树是以自下而上的方式构建的，直到我们最终得到一组根节点。树索引从一组节点（成为该树中的叶节点）构建一个层次树、查询树索引涉及从根节点向下遍历到叶节点。默认情况子节点设置（child\_branch\_factor=1），查询在给定父节点的情况下选择一个子节点。如果（child\_branch\_factor=2），查询会在每个级别选择两个子节点。Tree Index在查询时生成embedding，当然如果在查询时指定retriever\_mode=“embedding”，那么embedding将会延迟生成并被缓存起来。



需要在查询期间构建Tree Index，我们还需要向查询引擎添加retrier\_mode和response\_mode，并将GPTTreeIndex中的build\_Tree参数设置为False。

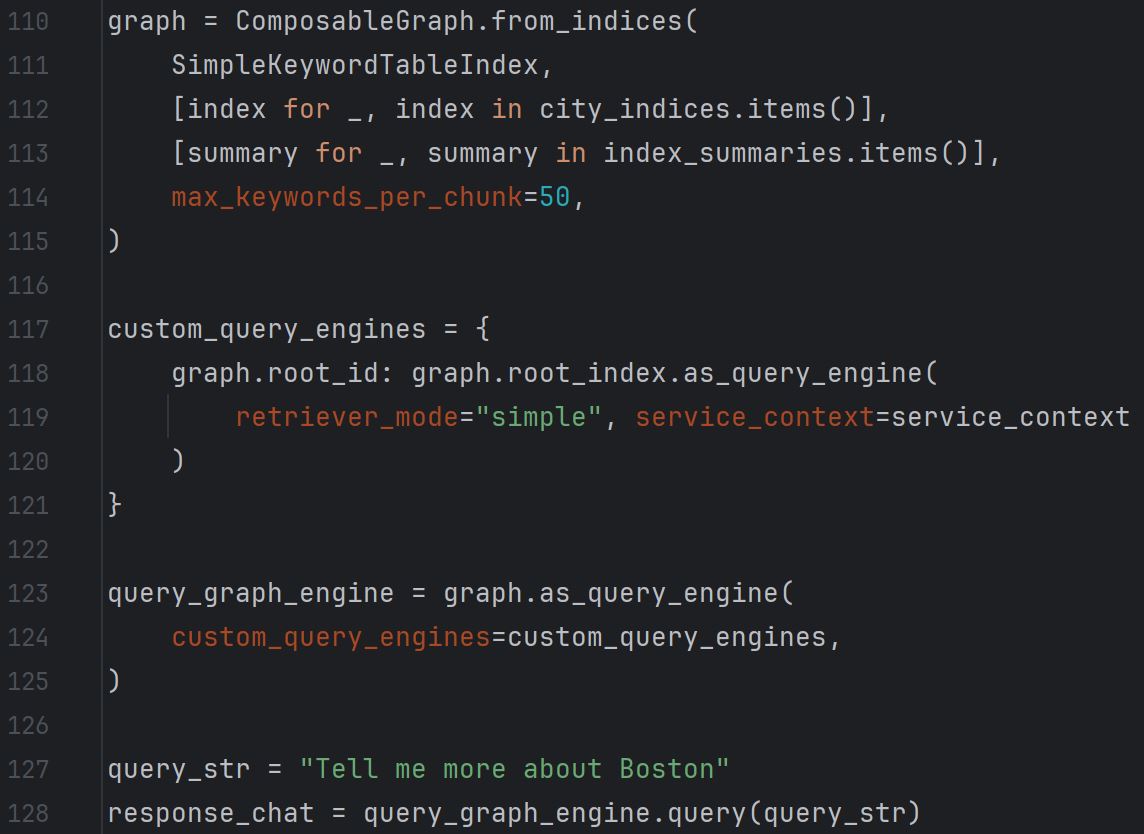
### **可组合性图索引（Composability Graph Index）**

LlamaIndex通过在现有Index的基础上composite indices，此功能能高效地索引完整的文档层次结构，并为GPT提供量身定制的知识。通过利用可组合性，可以在多个级别定义索引，例如单个文档的较低级别索引和文档组的较高级别索引。举例：

* 可以为每个文档中的文本创建一个树索引。
* 生成一个列表索引，该索引覆盖整个文档集合的所有树索引。

下面通过一个实例来说明一下可组合性图索引的能力：

* 从多个文档创建树索引
* 从树索引生成摘要。（树索引对于汇总文档集合非常有用）
* 我们将创建一个可组合性图索引（Graph），它在3个树索引之上有一个可组合性图索引索引。
* 最后查询图索引（Graph）



### **知识图谱索引（Knowledge Graph Index）**

知识图谱索引通过在一组文档上提取形式为（主语、谓语、宾语）的知识三元组来构建索引，在查询期间，可以只使用知识图作为上下文进行查询，也可以利用每个实体底层文本作为上下文。通过利用底层文本，我们可针对文档的内容提出更复杂的查询。在RAG中知识图谱的价值，如一个查询问题涉及到的上下文分布在一本书的30页分块的时候，传统的“分割数据，Embedding后再向量搜索”方法在多个文档块里用TOP-K去搜索的方法很难得到这种分散细粒的完整信息。而且，这种方法还很容易遗漏互相关联的文档块，从而导致信息检索不完整。知识图谱可减少基于嵌入的语义搜索所导致的不准确性。在保有领域知识的知识图谱则是直接可以缓解、消除这种幻觉的手段。

