基于Milvus和LangChain的LLM问答系统-AI法律助手设计与实现

基于Milvus和LangChain的LLM问答系统-AI法律助手设计与实现

本文内容已经发布在我的github主页。链接如下：
https://github.com/weisuzhou/software_homework

1.大语言模型的介绍

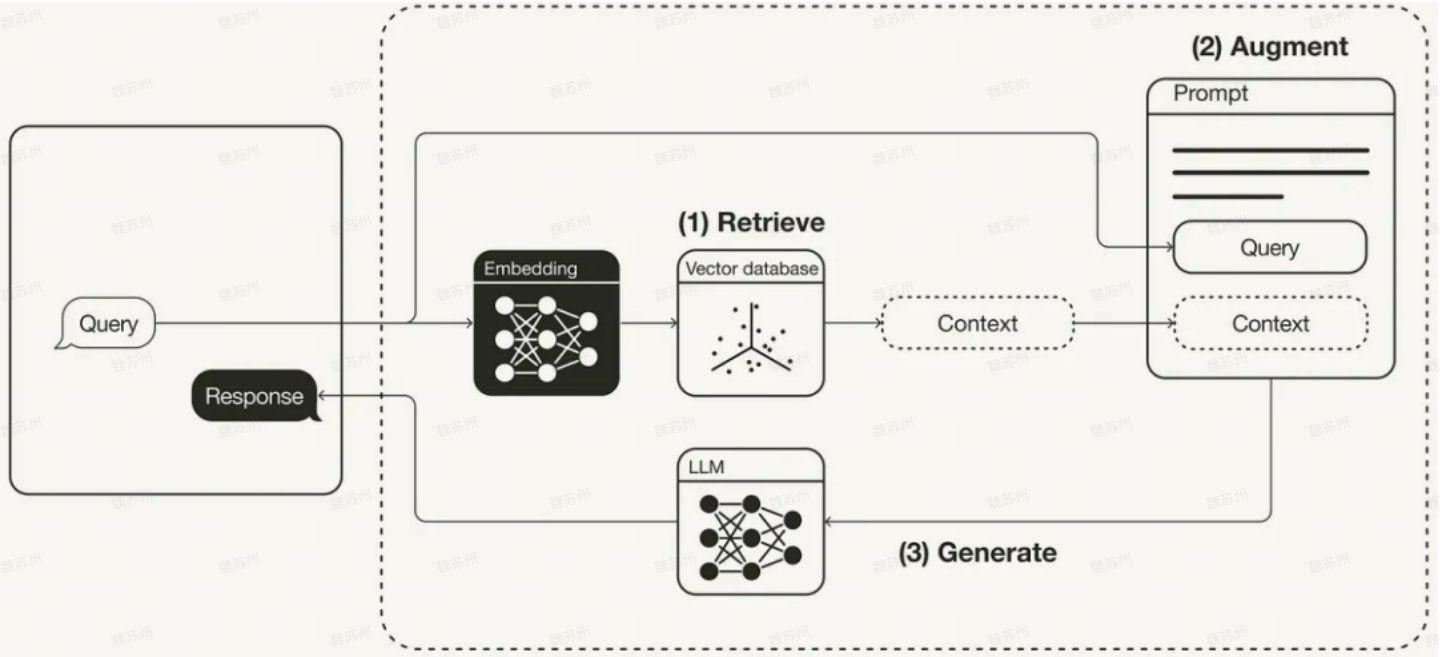
大语言模型（Large Language Models，简称LLMs）是指那些经过大量数据训练，拥有大量参数的机器学习模型，它们能够理解和生成自然语言。这些模型通常是基于深度学习技术构建的，特别是变换器（Transformer）架构，它在处理序列数据方面表现出色。大语言模型的特点是参数众多，可能包含数十亿甚至数千亿个参数，数据量大，为了训练这些模型，需要大量的文本数据，这些数据可以来自书籍、网站、论坛等。它们能够理解上下文信息，从而生成更加准确和相关的回答或内容，因此大语言模型通常能够在多种自然语言处理任务上表现出色，如文本分类、情感分析、机器翻译、摘要生成等。大语言模型在人工智能领域具有重要的应用价值，它们推动了自然语言处理技术的发展，并在多个领域内提供了创新的解决方案。

大语言模型的训练集基本都是构建于网络公开的数据，对于一些实时性的、非公开的或离线的数据是无法获取到的。虽然，大模型浪潮已经席卷了很多行业，但当涉及到行业细分领域时，通用大模型就会面临专业知识不足的问题。有时候大模型会一本正经地胡说八道，尤其是在大模型自身不具备某一方面的知识或不擅长的场景。而这种幻觉问题的区分是比较困难的，因为它要求使用者自身具备相应领域的知识。相对于成本昂贵的“Post Train”或“SFT”，基于RAG的技术方案往往成为一种更优选择。利用RAG架构，通过检索获取相关的知识并将其融入Prompt，让大模型能够参考相应的知识从而给出合理回答。

2.RAG检索增强生成

为了大模型的回答在某一领域更准确、更专业和更具时效性，检索增强生成的技术（RAG）被提出，并受到学术界和工业界的广泛关注。RAG 是一种相对较新的人工智能技术，可以通过允许大型语言模型 (LLM) 在无需重新训练的情况下利用额外的数据资源来提高生成式 AI 的质量。RAG 模型基于组织自身的数据构建知识存储库，并且存储库可以不断更新，以帮助生成式 AI 提供及时的上下文答案。实施 RAG 需要矢量数据库等技术，这些技术可以快速编码新数据，并搜索该数据以输入给LLM模型。

检索增强生成RAG其整体可划分为两个阶段：检索阶段和生成阶段。在检索阶段，算法搜索并检索与用户输入相关的信息片段；在生成阶段，大模型依据增强的提示和预训练的内部表示中提取信息，为用户量身定做生成答案，将答案发送给用户。其一般的工作流程如图所示。



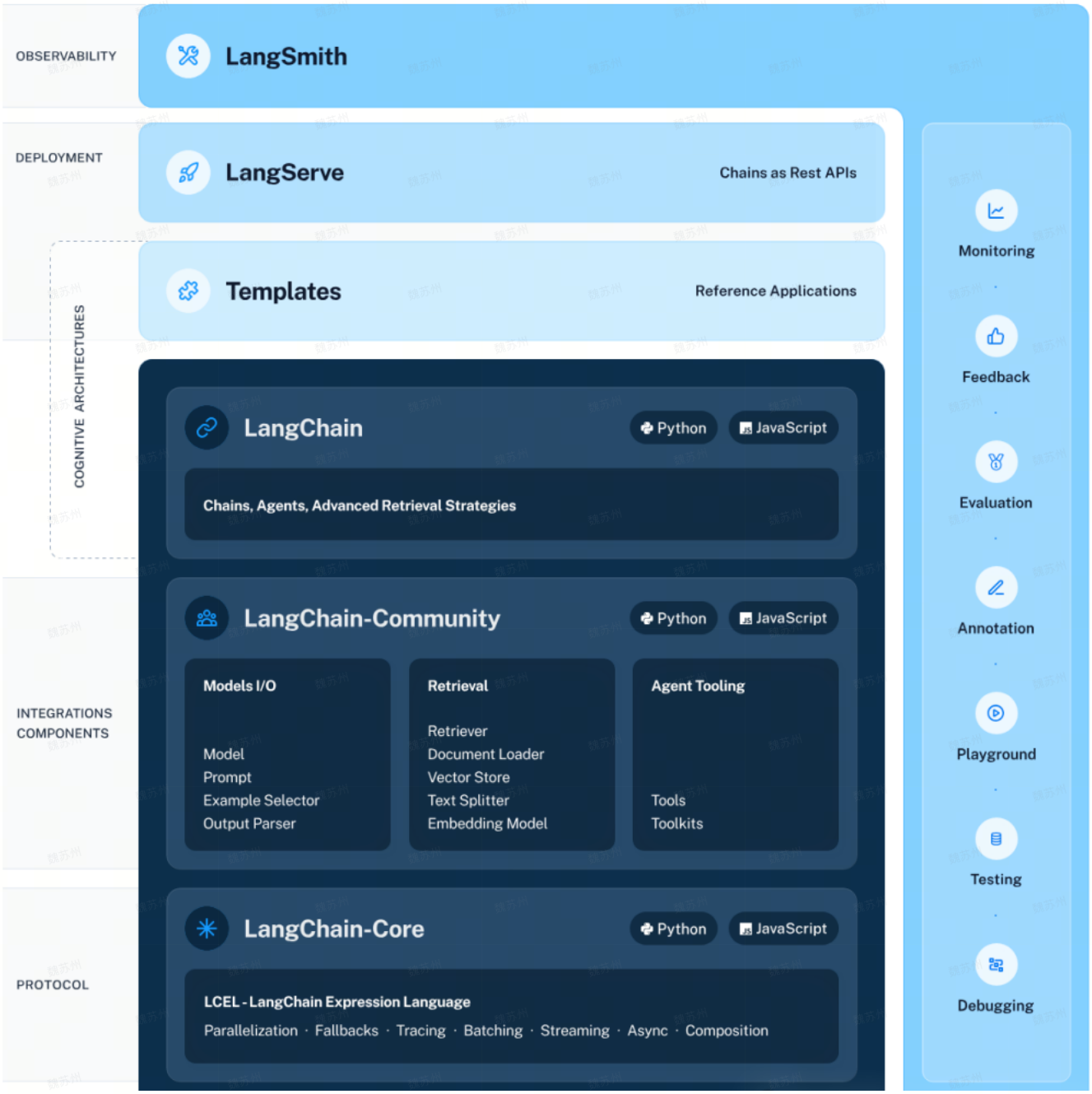
- 检索(Retrive)：根据用户请求从外部知识源检索相关上下文。为此，使用嵌入模型将用户查询嵌入到与向量数据库中的附加上下文相同的向量空间中。这允许执行相似性搜索，并返回矢量数据库中最接近的前 k 个数据对象。
- 增强(Augment)：用户查询和检索到的附加上下文被填充到提示模板中。
- 生成(Generate)：检索增强提示被馈送到 LLM，生成内容并返回给用户。

RAG的核心理解为“检索+生成”，前者主要是利用向量数据库的高效存储和检索能力，召回目标知识；后者则是利用大模型和Prompt工程，将召回的知识合理利用，生成目标答案。

3.LangChain的介绍

LangChain本质上是一个用于开发由LLM驱动的应用程序的框架。它通过连接LLM和外部一切可以利用的能力，给应用程序赋能，使得应用程序能够具备上下文意识和推理能力。LangChain 允许应用程序将语言模型连接到上下文来源，如提示指令、示例或需要回应的内容，从而增强模型的理解和生成能力。LangChain 支持将大型语言模型连接到私有数据源，如数据库或文件，从而允许模型从这些数据中提取信息。LangChain 的设计目标是为了实现数据感知和主动性的应用程序，它通过提供工具和抽象，使得开发者可以构建能够与环境交互并从数据源中学习和推理的智能应用程序。LangChain包括语言模型包装器、提示模板、索引、链和代理等核心概念，这些组件共同工作以解决特定的语言处理任务。

整个LangChain由六层组成，架构图如下。

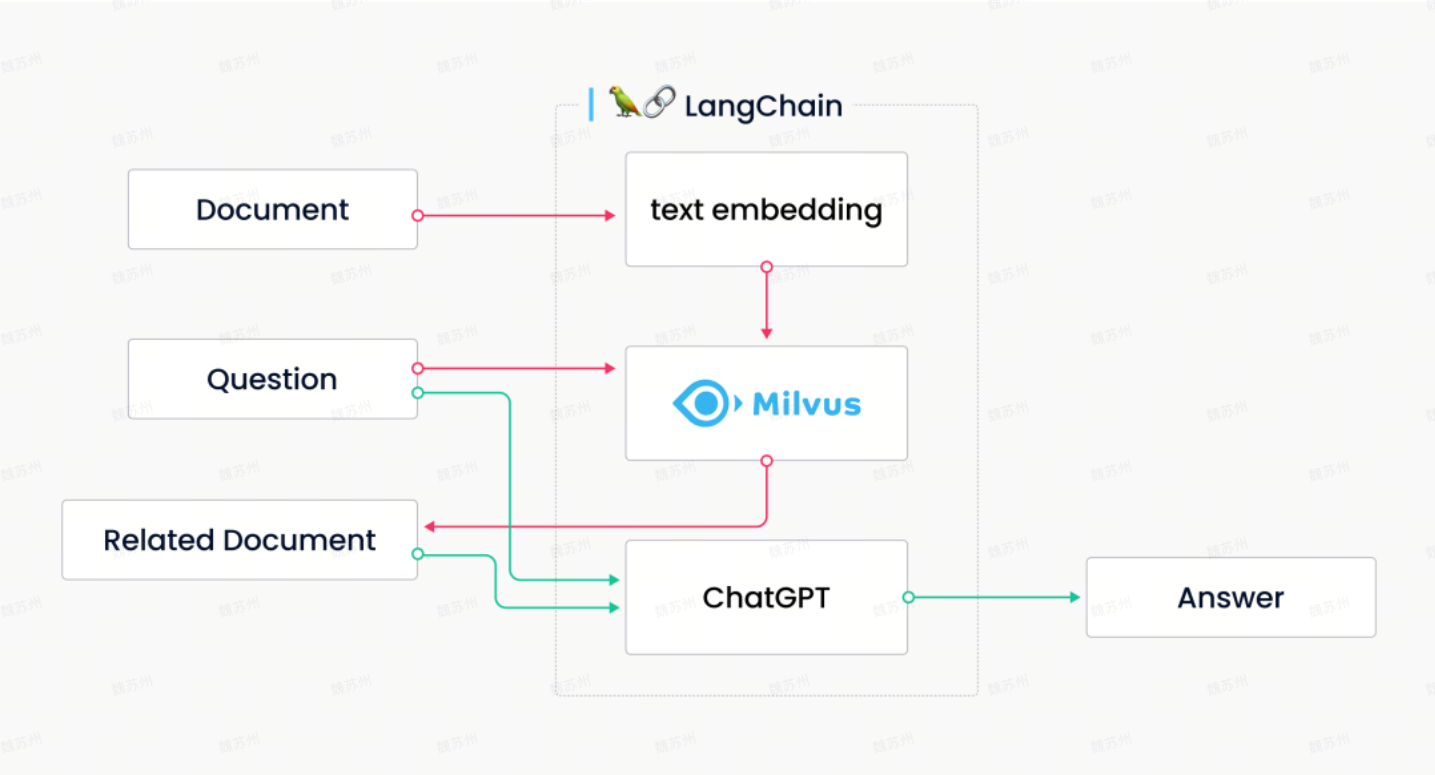


- 最底层是协议层，包括LangChain表达语言(LCEL)和一些基础抽象类。主要是提示词模板、输出解析器、链的抽象类，通过LCEL语法以及管道的概念，来简化代码的开发。
- 第二层是集成组件层，包括各种大语言模型输入输出、外部知识库检索、智能体工具等第三方集成。具体的比如智谱AI大语言模型包装类、嵌入模型HuggingFaceEmbeddings、向量数据库FAISS等，都在这一层。
- 第三层是认知架构层，包括构成应用程序认知架构的链、智能体和检索策略。主要是链(Chains)、智能体(Agents)和检索策略的具体实现类。
- 第四层是应用模板层，包括各种参考应用模板。多种可以快速运行部署的应用示例，参考价值比较高。
- 第五层是部署层，LangServe，是一个用于将LangChain部署为REST API的库。
- 第六层是表现层，LangSmith，一个开发平台，允许调试、测试、评估和监控LangChain应用。

LangChain 设计如下：

- Data-aware：连接 LLM 与其他数据源。
- Agentic：允许 LLM 与 LangChain 环境交互。

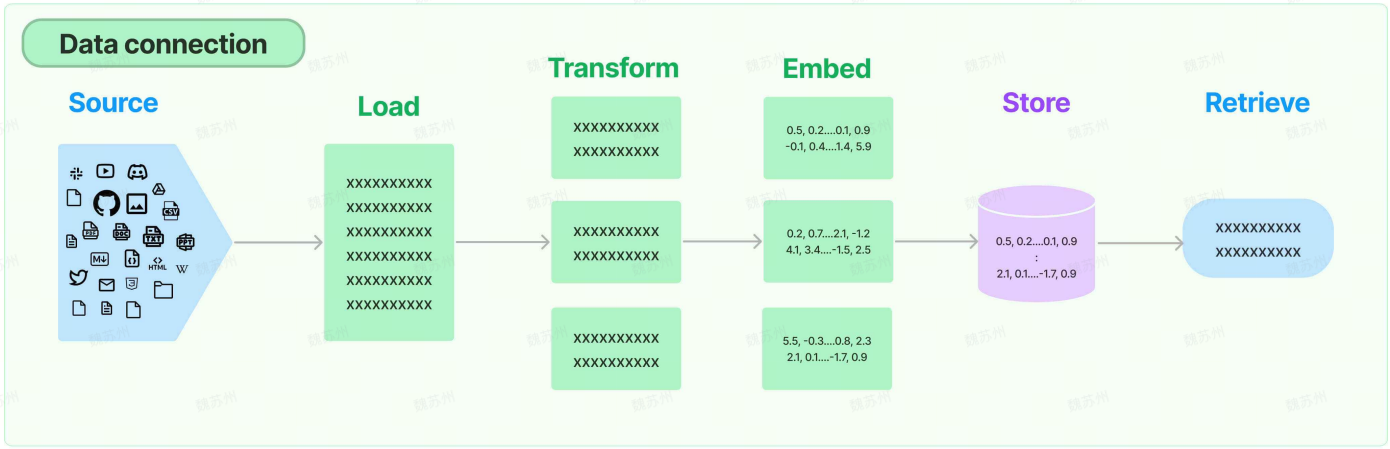
LangChain 包括许多模块，例如 Models、Prompts、Memory、Indexes、Chains、Agents 和 Callbacks。对于每个模块，LangChain 都提供标准化的可扩展接口。LangChain 还支持外部集成，甚至可实现端到端。LLM Wrapper 是 LangChain 的核心功能，提供了许多 LLM 模型，例如 OpenAI、Cohere、Hugging Face 等模型。LangChain 提供一系列有用的大型语言模型（LLMs），可满足多样的用户需求。LangChain 的另一大亮点是其扩展功能——集成各种向量数据库（如 Milvus、Faiss 等），因此可以很好地进行语义搜索。LangChain 通过 VectorStore Wrapper 提供了标准化的接口，从而简化数据加载和检索的流程。例如，大家可以使用 LangChain 的 Milvus 类，通过 `from_text` 方法存储文档的特征向量，然后调用 `similarity_search` 方法获取查询语句的相似向量（也就是在向量空间中找到距离最接近的文档向量），从而轻松实现语义搜索。工作流程如下图所示。



首先，在 Milvus 中存储由官方文档转化而来的文本向量。然后，在响应问题时搜索相关文档（如上图中红色箭头流程所示）。ChatGPT 最后根据正确的上下文回答问题，从而产生准确的答案（如上图中绿色箭头流程所示）。上述事例说明，LangChain + Milvus的组合可实现文本存储，用户无需标记、训练数据或进行额外开发和微调，只需将文本数据转化为向量并存储在 Milvus 中，即可解决幻觉问题，大大提高回答的准确性。

4.LangChain中RAG的使用

LangChain中提供了RAG应用程序的所有构建模块。如下图所示。



- 文档加载器：文档加载器用来加载各种不同类型的文档。LangChain提供了100 多种不同的文档加载器，可以加载各种类型文档，包括：CSV、HTML、JSON、Markdown、PDF、DOC、XLS、图片、视频、音频等等。具体的文档加载器参考：https://python.langchain.com/docs/integrations/document_loaders。
- 文本分割器：加载文档后，对于长文档，需要分割成更小的块，以适合LLM的上下文窗口。LangChain 有许多内置的文档转换器，可以轻松地拆分、组合、过滤和以其他方式操作文档。LangChain 提供了多种不同类型的文本分割器。如下图表所示。类型：文本分割器的类型；分割依据：此文本拆分器如何拆分文本；添加元数据：此文本拆分器是否添加有关每个块来自何处的元数据。描述：分离器的描述，包括有关何时使用它的建议。

类型	分割依据	添加元数据	描述
Recursive	用户定义的字符列表		递归地分割文本。递归地分割文本的目的是尝试使相关的文本片段彼此相邻。这是开始分割文本的推荐方法。
HTML	HTML 特定字符	✅	根据 HTML 特定字符分割文本。值得注意的是，这添加了有关该块来自何处的相关信息（基于 HTML）
Markdown	Markdown 特定字符	✅	根据 Markdown 特定字符分割文本。值得注意的是，这添加了有关该块来自何处的相关信息（基于 Markdown）
Code	代码（Python、JS）特定字符		根据特定于编码语言的字符分割文本。有 15 种不同的语言可供选择。
Token	Tokens		基于tokens分割文本。有几种不同的方法来衡量tokens.
Character	用户定义的字符		根据用户定义的字符分割文本。比较简单的方法之一。
[Experimental] Semantic Chunker	句子		首先对句子进行分割。然后，如果它们在语义上足够相似，则将它们相邻地组合起来。

- 嵌入模型

嵌入（Embedding）是一种将单词、短语或整个文档转换为密集向量的技术。每个单词或短语被转换成一组数字，这组数字捕捉了该文本的某些语义特征。嵌入模型通过将文本转换为计算机可以处理的

数值形式（即向量），使得计算机能够理解和处理自然语言。通过嵌入捕获文本的语义，可以快速有效地找到文本的其他相似部分。LangChain提供多种嵌入模型和方法，参考：https://python.langchain.com/docs/integrations/text_embedding。

- 向量数据库

有了嵌入模型对数据的向量化，就需要数据库来支持这些嵌入数据的高效存储和搜索。LangChain提供了50多种不同向量数据库的集成，可以参考：https://python.langchain.com/docs/integrations/vectorstores，目前Milvus评分最高。

5.环境准备

在使用LangChain进行AI应用程序开发前，需要准备好相应的开发环境，包括Conda、Jupyter Notebook、使用的AI大模型。

5.1安装Conda

Conda是一个Python库的安装、环境的管理的工具。目前有AnaConda和MiniConda两种，都是Continuum Analytics的开源项目。这两种的区别就是：AnaConda大而全，预装了大部分科学计算库(99%其实用不上)，提供了图形界面，适合初学者；MiniConda小而精，只有命令行。个人推荐使用MiniConda，比较轻量，节省时间和空间，需要什么再安装就可以。MiniConda的下载地址如下：https://docs.conda.io/projects/miniconda/en/latest/ ，windows和mac版本都包括了，下载好直接安装即可。

创建虚拟环境：`conda create -n langchain python=3.9`

激活虚拟环境：`conda activate langchain`

5.2安装Jupyter Notebook

为了能更方便编程和调试，及时执行代码块，我们使用Jupyter Notebook来作为Python集成开发工具。使用Jupyter的优点是可以逐行执行代码，方便调试和排错，对新手友好。同时，也方便学习和阅读。

安装Jupyter Notebook：`conda install jupyter notebook`

启动Jupyter notebook：`jupyter notebook`

5.3安装LangChain

LangChain 库：提供了Python和JavaScript库，包含组件的接口和集成，以及现成的链和代理的实现。我们这里安装LangChain包。

安装LangChain：`pip install --upgrade langchain`

5.4数据库 Milvus安装与部署

Milvus是一个开源的向量数据库，专为大规模相似度搜索应用场景而设计。它可以存储和查询数十亿甚至万亿维度的向量数据，并支持多种相似度搜索算法。Milvus可以存储和管理数十亿甚至万亿维度

的向量数据。

Milvus支持多种相似度搜索算法，例如：

- 内积
- 欧几里得距离
- 余弦距离
- 汉明距离

Milvus数据库可以通过多种方式安装，这里我选择使用Docker安装部署。使用Docker Compose安装Milvus Standalone。

- 下载YML文件：下载milvus-standalone-docker-compose。并将其保存为docker-compose。手动或使用以下命令创建Yml。`wget https://github.com/milvus-io/milvus/releases/download/v2.2.11/milvus-standalone-docker-compose.yml -O docker-compose.yml`
- 开始Milvus：在与docker-compose相同的目录下。运行以下命令启动Milvus: `docker-compose up -d`
- 连接到Milvus：验证Milvus服务器正在侦听哪个本地端口。将容器名称替换为您自己的名称。`docker port milvus-standalone 19530/tcp`，可以通过该命令返回的本地IP地址和端口号连接到Milvus集群。
- 停止Milvus：要停止独立的Milvus，运行命令: `docker-compose down`，停止Milvus后删除数据，使用命令: `sudo rm -rf  volumes`

启动和测试

- 依赖安装：pymilvus>=2.1.0 hnswlib>=0.5.2 pybind11 milvus>=2.1.0
- 下载示例代码进行测试：下载并运行 hello_milvus.py，出现下图即说明向量数据库安装成功。`python hello_milvus.py`

```
=== start connecting to Milvus     ===

Does collection hello_milvus exist in Milvus: False

=== Create collection `hello_milvus` ===


=== Start inserting entities       ===

Number of entities in Milvus: 3000

=== Start Creating index IVF_FLAT  ===


=== Start loading                  ===


=== Start searching based on vector similarity ===

hit: id: 2998, distance: 0.0, entity: {'random': 0.9728033590489911}, random field: 0.9728033590489911
hit: id: 1262, distance: 0.08883658051490784, entity: {'random': 0.2978858685751561}, random field: 0.2978858685751561
hit: id: 1265, distance: 0.09590047597885132, entity: {'random': 0.3042039939240304}, random field: 0.3042039939240304
hit: id: 2999, distance: 0.0, entity: {'random': 0.02316334456872482}, random field: 0.02316334456872482
hit: id: 1580, distance: 0.05628091096878052, entity: {'random': 0.3855988746044062}, random field: 0.3855988746044062
hit: id: 2377, distance: 0.08096685260534286, entity: {'random': 0.8745922204004368}, random field: 0.8745922204004368
search latency = 0.1278s

=== Start querying with `random > 0.5` ===

query result:
-{'random': 0.6378742006852851, 'embeddings': [0.20963514, 0.39746657, 0.12019053, 0.6947492, 0.9535575, 0.5454552, 0.823
search latency = 0.0587s
query pagination(limit=4):
        [{'random': 0.6378742006852851, 'pk': '0'}, {'random': 0.5763523024650556, 'pk': '100'}, {'random': 0.942593589161
query pagination(offset=1, limit=3):
        [{'random': 0.5763523024650556, 'pk': '100'}, {'random': 0.9425935891639464, 'pk': '1000'}, {'random': 0.78932112
```

5.5安装pymilvus

pymilvus 是 Milvus 的 Python 客户端库，用于与 Milvus 服务器进行交互。使用pip命令安装pymilvus。

安装命令：`pip install pymilvus`

5.6准备LLM模型

LLM的选择有多种方案。这里我选择使用远程调用智谱AI的GLM-4的API，效果较好，token花费较低。

安装智谱的SDK包：`pip install zhipuai`

由于最新的LangChain 0.1.7集成的ChatZhipuAI类和最新zhipuai SDK版本兼容性方面有问题，需要重新包装一个类。代码如下：包装ChatZhipuAI类

创建调用的对话大模型对象，api key需要修改成你自己的：

`#填写您自己的APIKey`

`ZHIPUAI_API_KEY = "..."`

`llm = ChatZhipuAI(`

```
    temperature=0.1,
    api_key=ZHIPUAI_API_KEY,
    model_name="glm-4",
)
```

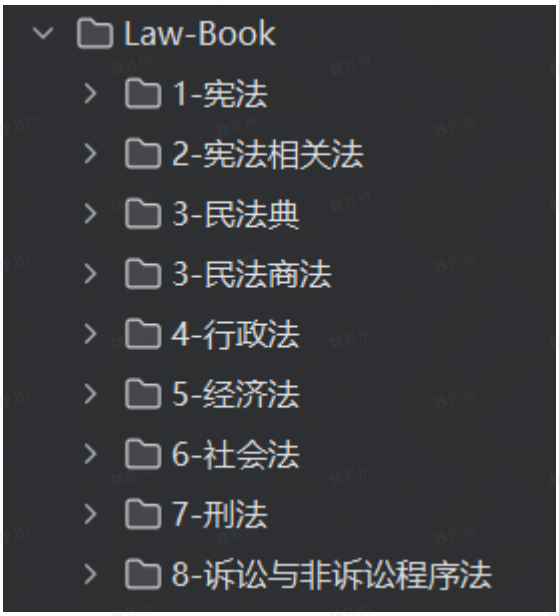
环境准备好之后，就可以开始使用LangChain进行AI应用程序开发了。LangChain其实就是通过各种链(Chain)将外部数据和计算连接到LLM，从而构建面向AI的应用程序。

6.开发案例-AI法律助手设计与实现

6.1加载文档

这里我找了一些关于中国法律的一些文献，包括中华人民共和国宪法、民法、行政法、经济法、社会法、刑法、诉讼与非诉讼程序法等，格式是.md文件，可以从以下链接获取：
https://nankai.feishu.cn/drive/folder/WAFKfM1qll9j9DdxSszci0itntd?from=from_copylink

文件夹内容是：



加载文档内容：

```
from langchain_community.document_loaders import DirectoryLoader
loader = DirectoryLoader( path: 'Law-Book', glob="**/*.md")
data = loader.load()
pprint(data)
```

```
[Document(page_content='中华人民共和国科学技术进步法\n\n1993年7月2日 第八届全国人民代表大会常务委员会第二次会议通过\n\n2007年12月29日 第十届全国人民代表大会常务委员会第三十一次会议第一次修订\n\n2021年12月
 Document(page_content='中国新民主主义革命的胜利和社会主义事业的成就，是中国共产党领导中国各族人民，在马克思列宁主义、毛泽东思想的指引下，坚持真理，修正错误，战胜许多艰难险阻而取得的。我国将长期处于社会主义初级阶
 Document(page_content='中华人民共和国乡村振兴促进法\n\n2021年4月29日 第十三届全国人民代表大会常务委员会第二十八次会议通过\n\n第一章\u3000总则\n\n第一条 为了全面实施乡村振兴战略，促进农业全面升级、农村全面
 Document(page_content='第十九条 国家维护国家基本经济制度和社会主义市场经济秩序，健全预防和化解经济安全风险的制度机制，保障关系国民经济命脉的重要行业和关键领域、重点产业、重大基础设施和重大建设项目以及其他重大经济
 Document(page_content='中华人民共和国森林法\n\n1984年9月20日 第六届全国人民代表大会常务委员会第七次会议通过\n\n1998年4月29日 第九届全国人民代表大会常务委员会第二次会议《关于修改〈中华人民共和国森林法〉的决
 Document(page_content='（二）已取得采矿权的矿山企业，因企业合并、分立，与他人合资、合作经营，或者因企业资产出售以及有其他变更企业资产产权的情形而需要变更采矿权主体的，经依法批准可以将采矿权转让他人采矿。\n\n前款
 Document(page_content='中华人民共和国环境保护法\n\n1989年12月26日 第七届全国人民代表大会常务委员会第十一次会议通过\n\n2014年4月24日 第十二届全国人民代表大会常务委员会第八次会议修订\n\n第一章 总则\n\n第一
 Document(page_content='中华人民共和国安全生产法\n\n2002年6月29日 第九届全国人民代表大会常务委员会第二十八次会议通过\n\n2009年8月27日第十一届全国人民代表大会常务委员会第十次会议《关于修改部分法律的决定》第
 Document(page_content='第五条 国家巩固和完善以家庭承包经营为基础、统分结合的双层经营体制，发展壮大农村集体所有制经济。\n\n第六条 国家建立健全城乡融合发展的体制机制和政策体系，推动城乡要素有序流动、平等交换和公
 Document(page_content='第二十七条 国家依法保护公民宗教信仰自由和正常宗教活动，坚持宗教独立自主自办的原则，防范、制止和依法惩治利用宗教名义进行危害国家安全的违法犯罪活动，反对境外势力干涉境内宗教事务，维护正常宗
```

6.2文本分割

通常可以使用递归字符文本分割器对文档对象进行分割，可以设置分割块字符的大小，和重合字符大小。

这里我分别设置为1000和200。

```
from langchain.text_splitter import RecursiveCharacterTextSplitter
text_splitter = RecursiveCharacterTextSplitter(
    chunk_size=1000, chunk_overlap=200, add_start_index=True
)
all_splits = text_splitter.split_documents(data)
pprint(len(all_splits))
```

6.3向量化存储

接下来我们首先使用嵌入模型对分割后的文档进行向量化，然后存储到向量数据库中。嵌入模型使用的是bge-large-zh-v1.5，这是BAAI北京智源人工智能研究院开源的，支持中、英文的嵌入模型，在开源嵌入模型中算是效果比较好的一个。向量数据库使用的是Milvus。

```
from langchain_community.embeddings import HuggingFaceBgeEmbeddings
model_name = "BAAI/bge-large-zh-v1.5"
model_kwargs = {"device": "cpu"}
encode_kwargs = {"normalize_embeddings": True}
bgeEmbeddings = HuggingFaceBgeEmbeddings(
    model_name=model_name, model_kwargs=model_kwargs, encode_kwargs=encode_kwargs
)
```

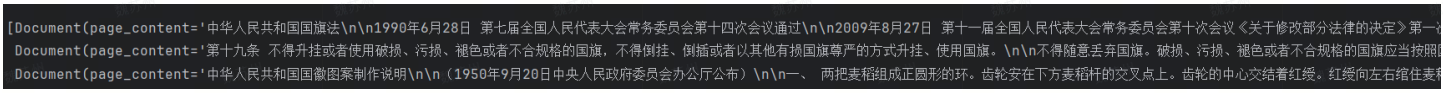
```
from langchain_community.vectorstores import Milvus
connection_args = {"host": "127.0.0.1", "port": "19530", "user": "root", "password": "123456"}
laws=Milvus(
    embedding_function=bgeEmbeddings,
    connection_args=connection_args,
    collection_name="laws"
)
```

6.4向量库检索

使用向量库进行检索。输入询问“国旗是？”，进行向量数据库检索。

```
from pprint import pprint
retriever = laws.as_retriever(search_type="similarity", search_kwargs={"k": 3})
query = "中华人民共和国的国旗是？"
docs =retriever.invoke(query)
pprint(docs)
```

检索结果。



6.5生成问答结果

使用检索链，串联向量库检索和大模型，根据用户的提问，生成问答结果。

```
from langchain_core.prompts import ChatPromptTemplate
from langchain_core.output_parsers import StrOutputParser
from langchain_core.runnables import RunnablePassthrough
prompt = ChatPromptTemplate.from_template("""仅根据所提供的上下文回答以下问题:
<context>
{context}
</context>
问题: {question}""")
retriever_chain = (
    {"context": retriever , "question": RunnablePassthrough()}
    | prompt
    | llm
    | StrOutputParser()
)
```

可以开始进行提问。例如输入“盗窃罪怎么处理？”

```
ans=retriever_chain.invoke("盗窃罪怎么处理？")
print(ans)
```

生成结果。

盗窃公私财物的，根据数额大小、次数和情节严重程度，分别处以不同的刑罚。具体规定如下：

1. 盗窃公私财物，数额较大的，或者多次盗窃、入户盗窃、携带凶器盗窃、扒窃的，处三年以下有期徒刑、拘役或者管制，并处或者单处罚金；

2. 数额巨大或者有其他严重情节的，处三年以上十年以下有期徒刑，并处罚金；

3. 数额特别巨大或者有其他特别严重情节的，处十年以上有期徒刑或者无期徒刑，并处罚金或者没收财产。

此外，还有以下几种情况：

1. 携带凶器抢夺的，依照刑法第二百六十三条的规定定罪处罚，即抢劫罪；

2. 聚众哄抢公私财物，数额较大或者有其他严重情节的，对首要分子和积极参加的，依照刑法第二百六十八条的规定定罪处罚；

3. 犯盗窃、诈骗、抢夺罪，为窝藏赃物、抗拒抓捕或者毁灭罪证而当场使用暴力或者以暴力相威胁的，依照刑法第二百六十三条的规定定罪处罚，即抢劫罪；

4. 将代为保管的他人财物非法占为己有，数额较大，拒不退还的，依照刑法第二百七十条的规定定罪处罚；

5. 将他人的遗忘物或者埋藏物非法占为己有，数额较大，拒不交出的，依照前款的规定处罚。

以上是关于盗窃罪的处理规定。如果刑法另有规定，则依照规定处理。

7.总结

本文通过介绍大语言模型、RAG技术、LangChain框架以及Milvus向量数据库，展示了如何结合这些技术来设计和实现一个AI法律助手应用程序。同时，本文还提供了环境准备和开发案例的具体步骤，以及相关的参考资源。希望本文的工作能为读者提供了一个完整的学习和实践指南。

参考链接

数据库 Milvus安装与部署：https://leavescn.com/Articles/Content/3160#google_vignette、https://cloud.tencent.com/developer/article/2338320

How to Build LLM Applications with LangChain Tutorial：https://www.datacamp.com/tutorial/how-to-build-llm-applications-with-langchain#what-are-large-language-models-(llms)?-large

用通俗易懂的方式讲解：一文讲清大模型 RAG 技术全流程：https://blog.csdn.net/2301_78285120/article/details/135494908

一文带你了解RAG(检索增强生成) | 概念理论介绍+ 代码实操（含源码）：https://cloud.tencent.com/developer/article/2373282

最新LangChain+GLM4开发AI应用程序系列（一）：快速入门篇：https://www.cnblogs.com/windpoplar/articles/18018719

解码 LangChain｜用 LangChain 和 Milvus 从零搭建 LLM 应用：https://cloud.tencent.com/developer/article/2317424?areaId=106001

最新LangChain+GLM4开发AI应用程序系列（三）：RAG检索增强生成篇：https://www.cnblogs.com/windpoplar/articles/18046618

通过阿里云Milvus和LangChain快速构建LLM问答系统：
https://help.aliyun.com/zh/emr/serverless-milvus/use-cases/use-milvus-and-langchain-to-build-the-llm-question-answering-system