🦜Langchain 0.2 RAG 快速入门 #1 环境配置

欢迎大家来参加 Langchain 0.2 RAG 快速入门课程!

我是**茉卷**。

在当今人工智能飞速发展的背景下，生成高度精确且与上下文相关信息的能力至关重要。

检索增强生成（Retrieval-Augmented Generation，简称RAG）是一种技术，通过整合外部知识源来提升生成模型的能力。这不仅提高了生成内容的质量，还确保其基于可靠的数据。

该系列教程致力于为您提供一份全面、循序渐进的指南，使用LangChain实现RAG。

LangChain是一个强大的框架，用于构建和部署大语言模型的应用程序。

我们从**基本的RAG流程**介绍开始，为理解如何将**基于检索的系统**与**生成模型**结合以产生准确且与上下文相关的响应奠定基础。

> - 数据索引
> - 文档召回
> - 内容生成

---

1 环境准备

- 大语言模型

- 嵌入模型
- Rerank 模型
- 向量数据库

1.1 langchain 相关包安装

环境： requirements.txt

```
langchain==0.2.6
langchain-chroma==0.1.2
langchain-community==0.2.6
langchain-core==0.2.11
langchain-experimental==0.0.62
langchain-openai==0.1.7
langchain-text-splitters==0.2.0
langchainhub==0.1.15
langgraph==0.1.5
langserve==0.2.2
langsmith==0.1.83
```

pip install -r requirements.txt

1.2 大语言模型准备

LLm: Glm4-air https://bigmodel.cn/dev/howuse/glm-4 （获得**Key**）

pip install zhipuai

通过 ChatOpenAI 生成一个 glm4 air 模型对象

```
from langchain_openai import ChatOpenAI

# 从我本地的一个文件，导入我设置的ZHIPU_AK
from ai_docs_cfg import ZHIPU_AK

glm4_air_model = ChatOpenAI(
    model_name="gLm-4-air",
    openai_api_base="https://open.bigmodel.cn/api/paas/v4",
```

```
    openai_api_key=ZHIPU_AK,

)
```

1.3 本地嵌入模型 和 本地Rerank 模型

安装适合自己电脑(GPU)的Torch (非常重要！！)

https://pytorch.org/get-started/locally/

嵌入模型： 文本 → 向量

Rerank 模型：根据某种算法，将文本根据相关度（针对某个问题），再排序

本地嵌入模型 ： 网易有道 BCEmbedding

https://github.com/netease-youdao/BCEmbedding

🍎 Model List

https://github.com/netease-youdao/BCEmbedding#-model-list

Model Name	Model Type	Languages	Parameters	Weights
bce-embedding-base_v1	`EmbeddingModel`	ch, en	279M	Huggingface, 国内通道
bce-reranker-base_v1	`RerankerModel`	ch, en, ja, ko	279M	Huggingface, 国内通道

配置本地BCE嵌入模型

```
from langchain_community.embeddings.huggingface import HuggingFaceEmbeddings

# 模型所在路径
embedding_model_name = 'D:\LLM\\bce_modesl\\bce-embedding-base_v1'

# 使用 GPU
embedding_model_kwargs = {'device': 'cuda:0'}
```

```
8
9 # 使用 CPU
10 # embedding_model_kwargs = {'device': 'cpu'}
11
12 # 配置
13 embedding_encode_kwargs = {'batch_size': 32, 'normalize_embeddings': True, }
14
15 # 初始化
16 embed_model = HuggingFaceEmbeddings(
17     model_name=embedding_model_name,
18     model_kwargs=embedding_model_kwargs,
19     encode_kwargs=embedding_encode_kwargs
20 )
21
```

使用本地BCE嵌入模型

```
1     # 小测试
2     query = '同学你好！'
3     passages = ['天真热', '谢谢！']
4     query_embedding = embed_model.embed_query(query)
5     passages_embeddings = embed_model.embed_documents(passages)
```

配置本地BCE Rerank 模型

pip install BCEmbedding==0.1.5

```
1 from BCEmbedding import RerankerModel
2
3 query = '你最喜欢的电影是什么？'
4 passages = [
5     '我最喜欢的电影是《阿甘正传》。',
6     '我最喜欢的电影类型是恐怖片',
7     '我最喜欢的电影导演是斯皮尔伯格。',
8     '我最喜欢的书是《哈利波特》。',
9     '我不喜欢看电影。',
10     '我喜欢看科幻电影，尤其是《星际穿越》。',
11     '我最喜欢的音乐是爵士乐。',
12     '昨天晚上我看了一部很好的电影。',
13     '我最喜欢的运动是篮球。',
14     '我喜欢吃披萨。'
15 ]
16
```

```
# construct sentence pairs
sentence_pairs = [[query, passage] for passage in passages]
reranker_model = 'D:\LLM\\bce_modesl\\bce-reranker-base_v1'

# 初始化 reranker model
model = RerankerModel(model_name_or_path=reranker_model)

# method 0: calculate scores of sentence pairs
scores = model.compute_score(sentence_pairs)

# method 1: rerank passages
rerank_results = model.rerank(query, passages)
doc_list = rerank_results['rerank_passages']
for line in doc_list:
    print(line)
pass

```

1.4 云端嵌入模型

```
from AI_docs.ai_docs_cfg import MY_QIANFAN_AK, MY_QIANFAN_SK
from langchain_community.embeddings.baidu_qianfan_endpoint import
QianfanEmbeddingsEndpoint
import os

os.environ["QIANFAN_AK"] = MY_QIANFAN_AK
os.environ["QIANFAN_SK"] = MY_QIANFAN_SK
# 英文
#embeddings_model = QianfanEmbeddingsEndpoint(model="bge_large_en",
endpoint="bge_large_en")

# 中文
embeddings_model = QianfanEmbeddingsEndpoint(model="bge_large_zh",
endpoint="bge_large_zh")

# 小测试
query = '同学你好！'
query_embedding = embeddings_model.embed_query(query)
print(len(query_embedding))
```

1.5 本地向量数据库 Chroma

```python
from langchain_community.embeddings.huggingface import HuggingFaceEmbeddings
from langchain_community.vectorstores import Chroma
from langchain_community.document_loaders import WebBaseLoader
from langchain.text_splitter import RecursiveCharacterTextSplitter

# 模型所在路径
embedding_model_name = 'D:\LLM\\bce_modesl\\bce-embedding-base_v1'

# 使用 GPU
embedding_model_kwargs = {'device': 'cuda:0'}

# 使用 CPU
# embedding_model_kwargs = {'device': 'cpu'}

# 配置
embedding_encode_kwargs = {'batch_size': 32, 'normalize_embeddings': True, }

# 初始化
embed_model = HuggingFaceEmbeddings(
    model_name=embedding_model_name,
    model_kwargs=embedding_model_kwargs,
    encode_kwargs=embedding_encode_kwargs
)

# Docs to index
urls = [
    "https://lilianweng.github.io/posts/2023-06-23-agent/",
]

# Load
docs = [WebBaseLoader(url).load() for url in urls]
docs_list = [item for sublist in docs for item in sublist]

# Split
text_splitter = RecursiveCharacterTextSplitter.from_tiktoken_encoder(
    chunk_size=500, chunk_overlap=0
)
doc_splits = text_splitter.split_documents(docs_list)

# Add to vectorstore
vectorstore = Chroma(persist_directory="D:\\LLM\\my_projects\\chroma_db",
embedding_function=embed_model,collection_name="rag-chroma")

# 只需要添加一次数据
```

```
vectorstore.add_documents(doc_splits)
```

完整代码

```
from BCEmbedding import RerankerModel
from langchain_openai import ChatOpenAI
# 从我本地的一个文件，导入我设置的ZHIPU_AK
from ai_docs_cfg import ZHIPU_AK


if __name__ == '__main__':

    glm4_air_model = ChatOpenAI(
        model_name="gLm-4-air",
        openai_api_base="https://open.bigmodel.cn/api/paas/v4",
        openai_api_key=ZHIPU_AK,
    )

    from langchain_community.embeddings.huggingface import HuggingFaceEmbeddings

    # 模型所在路径
    embedding_model_name = 'D:\LLM\\bce_modesl\\bce-embedding-base_v1'

    # 使用 GPU
    embedding_model_kwargs = {'device': 'cuda:0'}

    # 使用 CPU
    # embedding_model_kwargs = {'device': 'cpu'}

    # 配置
    embedding_encode_kwargs = {'batch_size': 32, 'normalize_embeddings': True,}

    # 初始化
    embed_model = HuggingFaceEmbeddings(
        model_name=embedding_model_name,
        model_kwargs=embedding_model_kwargs,
        encode_kwargs=embedding_encode_kwargs
    )

    # 小测试
    query = '同学你好！'
```

```
    passages = ['天真热', '谢谢！']
    query_embedding = embed_model.embed_query(query)
    passages_embeddings = embed_model.embed_documents(passages)

    query = '你最喜欢的电影是什么？'
    passages = [
        '我最喜欢的电影是《阿甘正传》。',
        '我最喜欢的电影类型是恐怖片',
        '我最喜欢的电影导演是斯皮尔伯格。',
        '我最喜欢的书是《哈利波特》。',
        '我不喜欢看电影。',
        '我喜欢看科幻电影，尤其是《星际穿越》。',
        '我最喜欢的音乐是爵士乐。',
        '昨天晚上我看了一部很好的电影。',
        '我最喜欢的运动是篮球。',
        '我喜欢吃披萨。'
    ]

    # construct sentence pairs
    sentence_pairs = [[query, passage] for passage in passages]
    reranker_model = 'D:\LLM\\bce_modesl\\bce-reranker-base_v1'

    # 初始化 reranker model
    model = RerankerModel(model_name_or_path=reranker_model)

    # method 0: calculate scores of sentence pairs
    scores = model.compute_score(sentence_pairs)

    # method 1: rerank passages
    rerank_results = model.rerank(query, passages)
    doc_list = rerank_results['rerank_passages']
    for line in doc_list:
        print(line)
    pass

    from AI_docs.ai_docs_cfg import MY_QIANFAN_AK, MY_QIANFAN_SK
    from langchain_community.embeddings.baidu_qianfan_endpoint import
QianfanEmbeddingsEndpoint
    import os

    os.environ["QIANFAN_AK"] = MY_QIANFAN_AK
    os.environ["QIANFAN_SK"] = MY_QIANFAN_SK
    # 英文
    # embeddings_model = QianfanEmbeddingsEndpoint(model="bge_large_en",
endpoint="bge_large_en")

    # 中文
```

```
    embeddings_model = QianfanEmbeddingsEndpoint(model="bge_large_zh", endpoint="bge_large_zh")

    # 小测试
    query = '同学你好！'
    query_embedding = embeddings_model.embed_query(query)
    print(len(query_embedding))

    from langchain_community.embeddings.huggingface import HuggingFaceEmbeddings
    from langchain_community.vectorstores import Chroma
    from langchain_community.document_loaders import WebBaseLoader
    from langchain.text_splitter import RecursiveCharacterTextSplitter

    # 模型所在路径
    embedding_model_name = 'D:\LLM\\bce_modesl\\bce-embedding-base_v1'

    # 使用 GPU
    embedding_model_kwargs = {'device': 'cuda:0'}

    # 使用 CPU
    # embedding_model_kwargs = {'device': 'cpu'}

    # 配置
    embedding_encode_kwargs = {'batch_size': 32, 'normalize_embeddings': True, }

    # 初始化
    embed_model = HuggingFaceEmbeddings(
        model_name=embedding_model_name,
        model_kwargs=embedding_model_kwargs,
        encode_kwargs=embedding_encode_kwargs
    )

    # Docs to index
    urls = [
        "https://lilianweng.github.io/posts/2023-06-23-agent/",
    ]

    # Load
    docs = [WebBaseLoader(url).load() for url in urls]
    docs_list = [item for sublist in docs for item in sublist]

    # Split
    text_splitter = RecursiveCharacterTextSplitter.from_tiktoken_encoder(
        chunk_size=500, chunk_overlap=0
    )
    doc_splits = text_splitter.split_documents(docs_list)
```

```
128
129     # Add to vectorstore
130     vectorstore = Chroma(persist_directory="D:\\LLM\\my_projects\\chroma_db",
    embedding_function=embed_model,
131                          collection_name="rag-chroma")
132
133     # 只需要添加一次数据
134     vectorstore.add_documents(doc_splits)
```