



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

```
1 langchain==0.2.6
2 langchain-chroma==0.1.2
3 langchain-community==0.2.6
4 langchain-core==0.2.11
5 langchain-experimental==0.0.62
6 langchain-openai==0.1.7
7 langchain-text-splitters==0.2.0
8 langchainhub==0.1.15
9 langgraph==0.1.5
10 langserve==0.2.2
11 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

```
1 from langchain_openai import ChatOpenAI
2
3 # 从我本地的一个文件, 导入我设置的ZHIPU_AK
4 from ai_docs_cfg import ZHIPU_AK
5
6 glm4_air_model = ChatOpenAI(
7     model_name="gLM-4-air",
8     openai_api_base="https://open.bigmodel.cn/api/paas/v4",
```

```
9     openai_api_key=ZHIPU_AK,  
10  
11 )
```

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 BCEembedding

<https://github.com/netease-youdao/BCEembedding>

🍏 Model List

<https://github.com/netease-youdao/BCEembedding#-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

```
1 from langchain_community.embeddings.huggingface import HuggingFaceEmbeddings  
2  
3 # 模型所在路径  
4 embedding_model_name = 'D:\\LLM\\bce_models\\bce-embedding-base_v1'  
5  
6 # 使用 GPU  
7 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 BCEEmbedding==0.1.5

```
1 from BCEEmbedding 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
```

```

17 # construct sentence pairs
18 sentence_pairs = [[query, passage] for passage in passages]
19 reranker_model = 'D:\LLM\bce_modesl\bce-reranker-base_v1'
20
21 # 初始化 reranker model
22 model = RerankerModel(model_name_or_path=reranker_model)
23
24 # method 0: calculate scores of sentence pairs
25 scores = model.compute_score(sentence_pairs)
26
27 # method 1: rerank passages
28 rerank_results = model.rerank(query, passages)
29 doc_list = rerank_results['rerank_passages']
30 for line in doc_list:
31     print(line)
32 pass
33

```

1.4 云端嵌入模型

```

1 from AI_docs.ai_docs_cfg import MY_QIANFAN_AK, MY_QIANFAN_SK
2 from langchain_community.embeddings.baidu_qianfan_endpoint import
  QianfanEmbeddingsEndpoint
3 import os
4
5 os.environ["QIANFAN_AK"] = MY_QIANFAN_AK
6 os.environ["QIANFAN_SK"] = MY_QIANFAN_SK
7 # 英文
8 embeddings_model = QianfanEmbeddingsEndpoint(model="bge_large_en",
  endpoint="bge_large_en")
9
10 # 中文
11 embeddings_model = QianfanEmbeddingsEndpoint(model="bge_large_zh",
  endpoint="bge_large_zh")
12
13 # 小测试
14 query = '同学你好!'
15 query_embedding = embeddings_model.embed_query(query)
16 print(len(query_embedding))

```

1.5 本地向量数据库 Chroma

```

1 from langchain_community.embeddings.huggingface import HuggingFaceEmbeddings
2 from langchain_community.vectorstores import Chroma
3 from langchain_community.document_loaders import WebBaseLoader
4 from langchain.text_splitter import RecursiveCharacterTextSplitter
5
6 # 模型所在路径
7 embedding_model_name = 'D:\\LLM\\bce_modesl\\bce-embedding-base_v1'
8
9 # 使用 GPU
10 embedding_model_kwargs = {'device': 'cuda:0'}
11
12 # 使用 CPU
13 # embedding_model_kwargs = {'device': 'cpu'}
14
15 # 配置
16 embedding_encode_kwargs = {'batch_size': 32, 'normalize_embeddings': True, }
17
18 # 初始化
19 embed_model = HuggingFaceEmbeddings(
20     model_name=embedding_model_name,
21     model_kwargs=embedding_model_kwargs,
22     encode_kwargs=embedding_encode_kwargs
23 )
24
25 # Docs to index
26 urls = [
27     "https://lilianweng.github.io/posts/2023-06-23-agent/",
28 ]
29
30 # Load
31 docs = [WebBaseLoader(url).load() for url in urls]
32 docs_list = [item for sublist in docs for item in sublist]
33
34 # Split
35 text_splitter = RecursiveCharacterTextSplitter.from_tiktoken_encoder(
36     chunk_size=500, chunk_overlap=0
37 )
38 doc_splits = text_splitter.split_documents(docs_list)
39
40 # Add to vectorstore
41 vectorstore = Chroma(persist_directory="D:\\LLM\\my_projects\\chroma_db",
42     embedding_function=embed_model, collection_name="rag-chroma")
43 # 只需要添加一次数据

```

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

完整代码

```
1 from BCEmbedding import RerankerModel
2 from langchain_openai import ChatOpenAI
3 # 从我本地的一个文件，导入我设置的ZHIPU_AK
4 from ai_docs_cfg import ZHIPU_AK
5
6
7 if __name__ == '__main__':
8
9     glm4_air_model = ChatOpenAI(
10         model_name="gLM-4-air",
11         openai_api_base="https://open.bigmodel.cn/api/paas/v4",
12         openai_api_key=ZHIPU_AK,
13     )
14
15     from langchain_community.embeddings.huggingface import
16     HuggingFaceEmbeddings
17     # 模型所在路径
18     embedding_model_name = 'D:\LLM\bce_modes\bce-embedding-base_v1'
19
20     # 使用 GPU
21     embedding_model_kwargs = {'device': 'cuda:0'}
22
23     # 使用 CPU
24     # embedding_model_kwargs = {'device': 'cpu'}
25
26     # 配置
27     embedding_encode_kwargs = {'batch_size': 32, 'normalize_embeddings': True,
28 }
29
30     # 初始化
31     embed_model = HuggingFaceEmbeddings(
32         model_name=embedding_model_name,
33         model_kwargs=embedding_model_kwargs,
34         encode_kwargs=embedding_encode_kwargs
35     )
36
37     # 小测试
38     query = '同学你好！'
```

```

38 passages = ['天真热', '谢谢! ']
39 query_embedding = embed_model.embed_query(query)
40 passages_embeddings = embed_model.embed_documents(passages)
41
42 query = '你最喜欢的电影是什么?'
43 passages = [
44     '我最喜欢的电影是《阿甘正传》。',
45     '我最喜欢的电影类型是恐怖片',
46     '我最喜欢的电影导演是斯皮尔伯格。',
47     '我最喜欢的书是《哈利波特》。',
48     '我不喜欢看电影。',
49     '我喜欢看科幻电影，尤其是《星际穿越》。',
50     '我最喜欢的音乐是爵士乐。',
51     '昨天晚上我看了一部很好的电影。',
52     '我最喜欢的运动是篮球。',
53     '我喜欢吃披萨。'
54 ]
55
56 # construct sentence pairs
57 sentence_pairs = [[query, passage] for passage in passages]
58 reranker_model = 'D:\\LLM\\bce_models\\bce-reranker-base_v1'
59
60 # 初始化 reranker model
61 model = RerankerModel(model_name_or_path=reranker_model)
62
63 # method 0: calculate scores of sentence pairs
64 scores = model.compute_score(sentence_pairs)
65
66 # method 1: rerank passages
67 rerank_results = model.rerank(query, passages)
68 doc_list = rerank_results['rerank_passages']
69 for line in doc_list:
70     print(line)
71 pass
72
73 from AI_docs.ai_docs_cfg import MY_QIANFAN_AK, MY_QIANFAN_SK
74 from langchain_community.embeddings.baidu_qianfan_endpoint import
QianfanEmbeddingsEndpoint
75 import os
76
77 os.environ["QIANFAN_AK"] = MY_QIANFAN_AK
78 os.environ["QIANFAN_SK"] = MY_QIANFAN_SK
79 # 英文
80 # embeddings_model = QianfanEmbeddingsEndpoint(model="bge_large_en",
endpoint="bge_large_en")
81
82 # 中文

```

```

83     embeddings_model = QianfanEmbeddingsEndpoint(model="bge_large_zh",
84     endpoint="bge_large_zh")
85     # 小测试
86     query = '同学你好!'
87     query_embedding = embeddings_model.embed_query(query)
88     print(len(query_embedding))
89
90     from langchain_community.embeddings.huggingface import
91     HuggingFaceEmbeddings
92     from langchain_community.vectorstores import Chroma
93     from langchain_community.document_loaders import WebBaseLoader
94     from langchain.text_splitter import RecursiveCharacterTextSplitter
95
96     # 模型所在路径
97     embedding_model_name = 'D:\LLM\bce_models\bce-embedding-base_v1'
98
99     # 使用 GPU
100     embedding_model_kwargs = {'device': 'cuda:0'}
101
102     # 使用 CPU
103     # embedding_model_kwargs = {'device': 'cpu'}
104
105     # 配置
106     embedding_encode_kwargs = {'batch_size': 32, 'normalize_embeddings': True,
107     }
108
109     # 初始化
110     embed_model = HuggingFaceEmbeddings(
111         model_name=embedding_model_name,
112         model_kwargs=embedding_model_kwargs,
113         encode_kwargs=embedding_encode_kwargs
114     )
115
116     # Docs to index
117     urls = [
118         "https://lilianweng.github.io/posts/2023-06-23-agent/",
119     ]
120
121     # Load
122     docs = [WebBaseLoader(url).load() for url in urls]
123     docs_list = [item for sublist in docs for item in sublist]
124
125     # Split
126     text_splitter = RecursiveCharacterTextSplitter.from_tiktoken_encoder(
127         chunk_size=500, chunk_overlap=0
128     )
129     doc_splits = text_splitter.split_documents(docs_list)

```

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