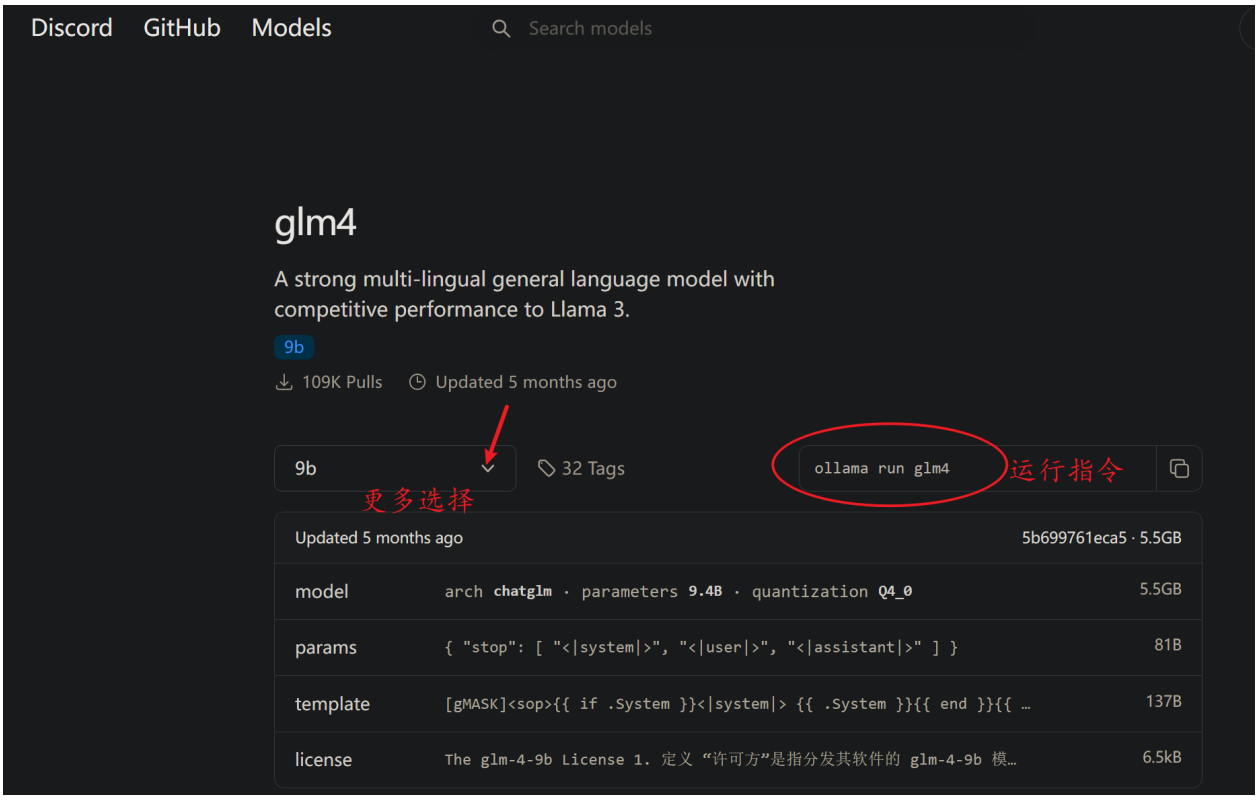
基于ollama的ragas评估指南

一、ollma环境部署

ollma对Windows环境支持比较好，安装运行也简单。当然在linux下会更灵活。安装后需要合理配置一些参数。

二、ollma模型下载

到模型库搜索关键词。找到目标模型后，可通过ollama pull 命令下载。注意至少一个语言模型和一个嵌入模型。



下载完成后，可以通过ollama list 命令显示当前可用的完整模型名称，这是在调用时必须提供的参数。

三、本地API调用

可以参考官方文档通过http请求进行最简单的实验，确保能正常工作。

ollama对openai风格接口已经部分兼容。部署好直接用openai库就可以调用。但为了便于ragas使用，我们用langchain_openai提供的包装进行测试。

```python
from langchain_openai import OpenAIEmbeddings, ChatOpenAI
from langchain_core.messages import HumanMessage, SystemMessage

# 创建OpenAI客户端实例，指向本地服务
llm = ChatOpenAI(  model="glm4:latest",
    base_url='http://127.0.0.1:11434/v1/',
    api_key="not-needed"  # API key可以是任意值，因为没有认证
)
messages = [
    SystemMessage(content="你是一个AI助手，善于解答各种问题"),
    HumanMessage(content="美联储进入降息周期后，对世界金融市场各个品种有哪些影响？")
]
response = llm.invoke(messages)
print(response.content)
one = OpenAIEmbeddings(model="aerok/xiaobu-embedding-v2:latest", deployment="aerok/xiaobu-embedding-v2:latest",
                       api_key='0', base_url='http://127.0.0.1:11434/v1/', )
out = one.embed_query(text="今天天气真热啊！")
print(out)
```

由于ollama对嵌入模型的openai风格接口仅支持输入文本而不支持输入token，所以需要修改langchain_openai 关于Embedding的源码。主要是 _get_len_safe_embeddings 和 _aget_len_safe_embeddings 两个方法。尤其是后者为异步调用方式，很重要。将源码中两个函数重命名或者注释掉，只保留其中最有价值的代码段就可以工作了。具体如下：

```python
    def _get_len_safe_embeddings(
        self, texts: List[str], *, engine: str, chunk_size: Optional[int] = None
    ) -> List[List[float]]:
        batched_embeddings: List[List[float]] = []
        for i, text in enumerate(texts,1):
            valid = text if len(text) <= 510 else text[-510:]
            response =  self.client.create(
                input= valid, **self._invocation_params           )
            if not isinstance(response, dict):
                response = response.model_dump()
            batched_embeddings.extend(r["embedding"] for r in response["data"])

        return  batched_embeddings

```

```
    async def _aget_len_safe_embeddings(
        self, texts: List[str], *, engine: str, chunk_size: Optional[int] = None
    ) -> List[List[float]]:
        batched_embeddings: List[List[float]] = []
        for i, text in enumerate(texts,1):
            valid = text if len(text) <= 510 else text[-510:]
            response = await  self.async_client.create(
                input= valid, **self._invocation_params      )
            if not isinstance(response, dict):
                response = response.model_dump()
            batched_embeddings.extend(r["embedding"] for r in response["data"])

        return  batched_embeddings
```

四、在ragas下的使用

具体代码如下：

```
import json
from langchain_openai import ChatOpenAI, OpenAIEmbeddings
from ragas import evaluate, RunConfig, EvaluationDataset
from ragas.llms import LangchainLLMWrapper
from ragas.embeddings import LangchainEmbeddingsWrapper
from ragas.metrics import (ContextPrecision, ContextRecall,

                           Faithfulness, AnswerRelevancy, SemanticSimilarity)

llm = ChatOpenAI( model="glm4:latest",
                base_url='http://127.0.0.1:11434/v1/',
                api_key="not-needed" )
evaluator_llm = LangchainLLMWrapper(llm)
embed_model = OpenAIEmbeddings(model="aerok/xiaobu-embedding-v2:latest",
                               deployment="aerok/xiaobu-embedding-v2:latest",
                               api_key='0', base_url='http://127.0.0.1:11434/v1/', )
evaluator_embeddings = LangchainEmbeddingsWrapper(embed_model)
metrics = [ Faithfulness(llm=evaluator_llm),
    AnswerRelevancy(llm=evaluator_llm, embeddings=evaluator_embeddings),
```

```
    ContextPrecision(llm=evaluator_llm),
    ContextRecall(llm=evaluator_llm),  ]

if __name__ == '__main__':
    # 读取json文件中的数据，最好是字典的列表。
    with (open('other/qainfo.json', 'r', encoding='utf-8')) as fr:
        qainfo = json.load(fr)['results']
        new = []
        for k in qainfo:
            one = {'user_input': k['query'],  'response': k['response'],
                   'reference': k['gt_answer'],
                   'retrieved_contexts': [one['text'] for one in k["retrieved_context"]]
                   }
            new.append(one)
    eval_dataset = EvaluationDataset.from_list(new[50:60]) # 选择需要评估的范围。
    print(eval_dataset)  # 答应评估数据集的基本信息
    results = evaluate(dataset=eval_dataset, metrics=metrics, batch_size=1,
                       run_config=RunConfig(timeout=900))
    print(results)
# {'context_recall': 0.860, 'factual_correctness': 0.498, 'faithfulness': 0.659}
    df = results.to_pandas()
    df.to_csv("result.csv", index=False)
```

其中数据集是参考ragas文档介绍准备的json格式。