**基于llama\_index的rag开发笔记**

1. **rag：**

简单来讲，rag就是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。因此，可以将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。

数据准备阶段：数据提取 >> 文本分割 >> 向量化（embedding) >> 数据入库

应用阶段：用户提问 >> 数据检索（召回） >> 注入Prompt >> LLM生成答案

1. **llama\_index：**
2. 开启控制台log输出

logging.basicConfig(stream=sys.stdout, level=logging.INFO)

logging.getLogger().addHandler(logging.StreamHandler(stream=sys.stdout))

llama\_index.core.set\_global\_handler("simple")

1. 设置openai账号与模型

os.environ["OPENAI\_API\_KEY"] = "xxxxxxxxx"

os.environ["OPENAI\_API\_BASE"] = "urlxxxxxxx"

llm = OpenAI(temperature=0, model="gpt-3.5-turbo", max\_tokens=2048)

Settings.llm = llm

1. 设置embedding模型

Settings.embed\_model = HuggingFaceEmbedding(model\_name="BAAI/bge-base-zh-v1.5")

1. 设置qa prompt与refine prompt

qa\_prompt = (

"We have provided context information below. \n"

"----------------\n"

"context:\n{context\_str}\n"

"----------------\n"

"你想要定义的prompt"

"问题:{query\_str}\n"

)

qa\_prompt\_tmpl = PromptTemplate(qa\_prompt, prompt\_type=PromptType.QUESTION\_ANSWER)

---------------

refine\_prompt\_tmpl = PromptTemplate(refine\_prompt, prompt\_type=PromptType.REFINE)

1. 自定义sentence splitter

class CustomSentenceSplitter(SentenceSplitter):

def \_split\_text(self, text: str, chunk\_size: int) -> List[str]:

return text.split("\n\n\n")

def split\_text(self, text: str) -> List[str]:

return text.split("\n\n\n")

1. 创建、存储、引用index

if not os.path.exists(storage\_path):

splitter = CustomSentenceSplitter()

documents = SimpleDirectoryReader("./data/").load\_data()

nodes = splitter.get\_nodes\_from\_documents(documents)

index = VectorStoreIndex(nodes)

index.storage\_context.persist(persist\_dir=storage\_path)

else:

storage\_context = StorageContext.from\_defaults(persist\_dir=storage\_path)

index = load\_index\_from\_storage(storage\_context)

1. 创建retriver与query engine

retriever = VectorIndexRetriever(

index=index,

node\_ids=list(index.index\_struct.nodes\_dict.values()),

callback\_manager=index.\_callback\_manager,

object\_map=index.\_object\_map,

similarity\_top\_k=5

)

query\_engine = RetrieverQueryEngine.from\_args(retriever,llm=llm, response\_mode="simple\_summarize")

1. 使用retriever进行检索

prod\_retrieve\_result = prod\_retriever.retrieve(question)

prod\_total\_result += f"产品编号{prod\_id}的产品文档内容: \n"

prod\_total\_result += "\n----------\n"

1. 使用query engine进行检索与回答

response = query\_engine.query(question)

response = Settings.llm.complete(pd2\_prompt)

1. 高级rag（检索路由、融合检索、关键词检索等）

通过精准检索或者目录检索得到对应内容或者index，再利用embedding后的问题进行检索，可以增加检索效率，减少误差空间。

将检索关键片段与文本长文一同加入prompt，融合两者使llm能够理解片段的详细信息与长文的上下关系。

先通过关键词检索到对应的index，再通过index进行二次检索，从而得到理想的内容。