RAG实战-2

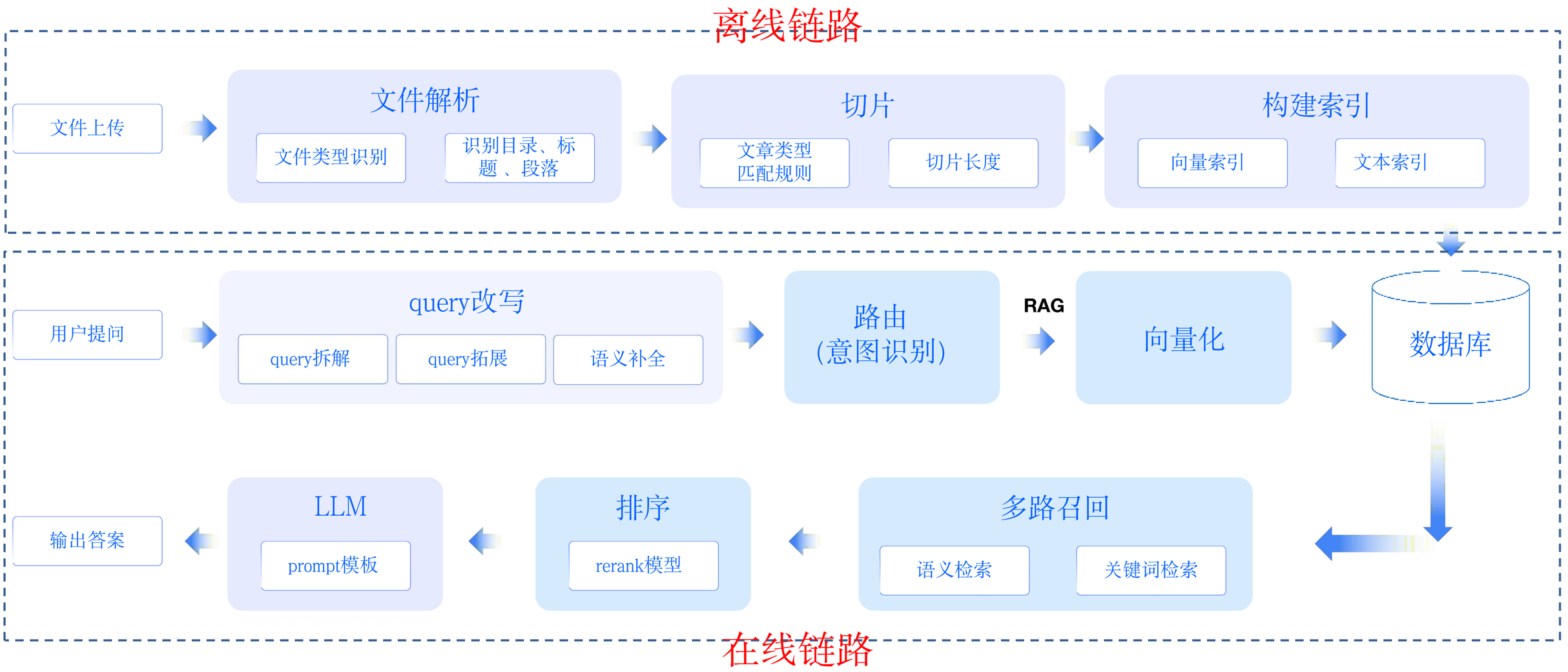
Project2 Review

1. 调用embedding模型，批次处理，不要每行调用，特别慢

2. 数据库的增删查改操作，尽量是面向对象的思路，而不是面向过程

3. 在线链路里，根据向量库返回的索引id，到mysql表中匹配对应的结果，不是直接读文件。

详细技术方案


离线链路
文件上传
文件解析
文件类型识别
识别目录、标题 、段落
切片
文章类型匹配规则
切片长度
构建索引
向量索引
文本索引
用户提问
query改写
query拆解
query拓展
语义补全
路由（意图识别）
RAG
向量化
数据库
输出答案
LLM
prompt模板
排序
rerank模型
多路召回
语义检索
关键词检索
在线链路

llama-index介绍

llama-index用于简化LLM与外部数据源之间的交互。

Data Loaders->Indexers(Embeddings)->(Vector Stores)->Retrievers（检索）->QueryEngine

1. Data Loaders：负责从不同的数据源中提取和加载文档（或其他非结构化数据）。

2. Indexers：负责从加载的文档中创建索引结构

3. Retrievers:检索器负责执行查询并从索引中检索相关文档。

4. QueryEngine:查询引擎负责根据用户输入的查询和检索到的文档+LLM，生成最终的答案或结果。

Data Loaders

Loaders	功能说明
`SimpleDirectoryReader`	LlamaIndex内置的，支持Markdown, PDFs, Word documents, Pc images, audio and video.
`WebPageReader`	加载网页内容并转为文档格式，支持从 URL 抓取数据。
`PandasCSVReader`	加载 CSV 文件并将其解析为文档，常用于表格数据处理。
`PDFReader`	提取 PDF 文件中的文本内容，支持多页文件解析。
`NotionDBReader`	从 Notion 数据库中提取数据，用于处理团队协作文档。
`DatabaseReader`	连接数据库并提取数据，支持 SQL 查询语句。
`GoogleDriveReader`	从 Google Drive 加载文件，支持多种文件格式（例如 Google Docs、F
`MarkdownReader`	加载 Markdown 文件并解析为结构化文档。

官网：https://docs.llamaindex.ai/en/stable/understanding/
组件可扩展：https://llamahub.ai

Indexers&Retrievers配套使用

Indexer	作用	Retriever	Retriever 作用	适用场景
VectorStoreIndex	将文档转化为向量进行索引，使用嵌入模型	VectorIndexRetriever	基于向量相似度的检索，使用预计算的文档向量	语义检索，基于向量的相似性检索，如问题回答
KeywordTableIndex	简单的关键词索引，适合小型数据集	KeywordTableSimpleRetriever	高效的小规模数据集的关键词检索	小规模文本数据的查询
DashScopeCloudIndex	文档的嵌入向量和索引信息会被存储在阿里云的云存储上	DashScopeCloudRetriever	同时支持了向量检索和文本检索	阿里云模型相关的向量检索
TreeIndex	用树形结构索引文档，适合层次结构的数据	TreeAllLeafRetriever	层次化数据检索，基于树的结构查找数据	有层级结构的数据（如分类、目录树等）
KnowledgeGraphIndex	构建知识图谱的索引结构	KGTableRetriever	用于从已经构建的知识图谱中查询数据，可以根据实体或关系进行查询，支持检索节点、边或子图。	知识图谱查询/关系推理

扩展

https://github.com/datawhalechina/tiny-universe/blob/main/README.md

1. 纯手工搭建 RAG 框架——Tiny RAG
2. 逐步预训练一个手搓大模型——Tiny LLM
3. ~~如何评估你的大模型——Tiny Eval~~
4. 深入剖析大模型原理——Qwen Blog
5. 手搓一个最小的 Agent 系统——Tiny Agent
6. 深入理解大模型基础——Tiny Transformer

扩展-2

1. 训练Tokenizer(分词器)：SentencePiece库来训练自定义的Tokenizer，从原始文本中学习词汇表。

2. 数据预处理：把文本数据->分词->token ID->保存为二进制文件。

3. 训练模型： Decoder only Transformer模型，token ID->embedding(高维向量)->Self Attention(调整，相似的词或子词在向量空间中更接近)

4. 使用模型生成文本：输入文本->分词->token ID->embedding->decoder（根据上下文更新向量）->预测下一个token在词汇表出现的置信度

大模型测评

1. 精确率（Precision）

在RAG中，精确率表示检索的结果有多“精确”。

定义：

$$\text{Precision} = \frac{\text{相关且被检索到的文档数量（TP）}}{\text{被检索到的总文档数量（TP + FP）}}$$

含义：

- 检索出的文档中，有多少是模型实际需要的。
- 在RAG场景中，这代表模型检索出的文档对生成结果有帮助的比例。例如，如果模型检索了10篇文档，其中7篇与用户问题相关，则精确率是70%。

场景中精确率的重要性：

- 如果精确率低，生成模型可能会基于错误的信息生成错误的回答。
- 适用于对结果质量有高要求的场景（如医疗或法律问答）。

大模型测评

2. 召回率（Recall）

在RAG中，召回率表示检索的全面性。

定义：

$$\text{Recall} = \frac{\text{相关且被检索到的文档数量（TP）}}{\text{所有相关文档的总数量（TP + FN）}}$$

含义：

- 在所有可能相关的文档中，有多少被成功检索到。
- 在RAG场景中，这代表模型是否找到了解决问题所需的所有信息。例如，假设某问题的答案需要5篇相关文档，而模型只检索到3篇，则召回率是60%。

场景中召回率的重要性：

- 如果召回率低，生成模型可能会缺乏关键信息，导致回答不完整。
- 适用于对信息覆盖率要求较高的场景（如技术文档生成）。

大模型测评

3. F1分数

F1分数是在RAG中用来平衡精确率和召回率的指标。

定义：

$$\mathrm{F1} = 2 \cdot \frac{\mathrm{Precision} \cdot \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}}$$

含义：

- F1分数是一种平衡指标，特别适用于需要同时兼顾检索精确性和全面性的场景。
- 在RAG中，F1分数可以衡量检索阶段的综合性能：既不希望检索出过多无关文档（降低生成准确性），也不希望遗漏关键文档（影响生成全面性）。

场景中F1分数的重要性：

- 适合权衡信息全面性与相关性之间的取舍。
- 在用户问题较复杂时（如涉及多个知识点的问题），较高的F1分数能有效提高生成结果的质量。

大模型测评

假设用户提问：“如何训练一个大语言模型？”

1. 相关文档（实际需要的）：5篇。

2. 检索到的文档：10篇
   * 3篇相关（TP=3）
   * 7篇不相关（FP=7）。

3. 漏检的相关文档：2篇（FN=2）。

调整向量库参数后，可以通过这几个指标评测优化效果。

计算：

- **精确率：**

$$\text{Precision} = \frac{3}{3+7} = 0.3\,(30\%)$$

- **召回率：**

$$\text{Recall} = \frac{3}{3+2} = 0.6\,(60\%)$$

- **F1分数：**

$$\text{F1} = 2 \cdot \frac{0.3 \cdot 0.6}{0.3+0.6} = 0.4\,(40\%)$$

大模型测评

AnswerRelevancyMetric

智能体答案相关性指标，通过智能体答案与用户提问的匹配程度。

高相关性的答案不仅要求模型能够理解用户的问题，还要求其能够生成与问题密切相关的答案。直接影响到用户的满意度和模型的实用性。

注意：此类方案是引入第三方大模型进行评测

参考方案：https://www.yuque.com/eosphoros/dbgpt-docs/czgl7bsfclc1xsmh

Project2-简化后的技术方案

在线链路（数据消费）：

用户输入查询
->向量化（input query -> embedding）
->向量库返回查询结果索引（返回匹配的索引ID）
->根据索引匹配到mysql数据表结果（根据ID查询数据表）why引入数据库？
->根据查询结果组装context上下文
->带入llm
->展示llm结果

Project2-可选部分

可选：把RAG能力封装为HTTP服务

1. 引入flask作为web框架，将聊天功能封装为http json请求
支持传参：
userInput:用户输入
historyMessages：历史对话内容

测试命令行：
curl -X POST http://localhost:8080/chat -H "Content-Type: application/json" -d '{"message": "鞋子有哪些尺码可以选择呢","historyMessages": ["历史聊天内容"]}'

Project2-实现关键点

1. faiss库查询匹配的索引
    ├——— faiss.read_index('XXX.index')
    ├——— faiss_read_index.search(将input转为embedding输入进行搜索)

2. mysql使用：
    ├——— 可以考虑使用轻量级ORM框架 peewee

Project2-实现关键点

```python
# models.py
from peewee import *
import atexit


# 创建数据库连接
db = MySQLDatabase( database: 'spring_boot_demo'
                   , user='root', password='', host='localhost')

# 定义模型
class AiContext(Model):  3 usages  9cc9
    id = AutoField()  # 主键使用 AutoField
    text = TextField()

    class Meta:  9cc9
        database = db
        table_name = 'ai_context'  # 映射表名

# 确保数据库连接
db.connect()

# 注册程序退出时关闭数据库连接的钩子
atexit.register(lambda: db.close())
```

AiContext.create() 用于插入新数据，它会自动生成 ID。

AiContext.select() 获取所有数据，或可以配合 .where() 添加过滤条件。
也可以使用AiContext.get()直接根据主键ID进行查询

AiContext.update() 用于更新数据，可以指定要更新的字段。

AiContext.delete() 用于删除数据，需指定删除条件。

Project2-控制台输出

```
用户： 鞋子有些什么尺码呢
助手： 根据提供的信息，鞋子的尺码范围在30-48之间。建议您参考提供的尺码表，并根据自己的脚长来选择合适的尺码。请注意，这里的尺码可能是指欧洲尺码系统。如果您需要更具体的指导，比如转换为国际通用尺码，可能需要更多的信息或参照具体品牌的尺码指南。
用户： 36码的脚可以买到合适的吗
助手： 根据提供的信息，您可以买到适合36码脚的鞋子，因为鞋子的尺码范围在30-48之间，所以36码在这个范围内。建议您参考提供的尺码表，根据自己的脚长选择合适的尺码以获得最佳的穿着体验。请注意，这里提到的尺码可能是按照欧洲尺码标准。如果您需要将其他尺码标准（如中国或美国尺码）转换为欧洲尺码，建议查看具体的尺码对照表。
用户： 鞋子有些什么品牌呢
助手： 提供的鞋款包括多个知名品牌，如Adidas、Nike、Puma、Reebok、New Balance、Asics、Under Armour等运动鞋品牌，还有Converse、Vans等休闲时尚品牌，以及Gucci、Balenciaga等高端奢侈品牌鞋款。您可以根据自己的需求和喜好选择合适的品牌和款式。
用户： 好的谢谢 我没有问题了
助手： 感谢您的咨询，再见！
==========conversation finish!
```