RAG评测方法与优化思路

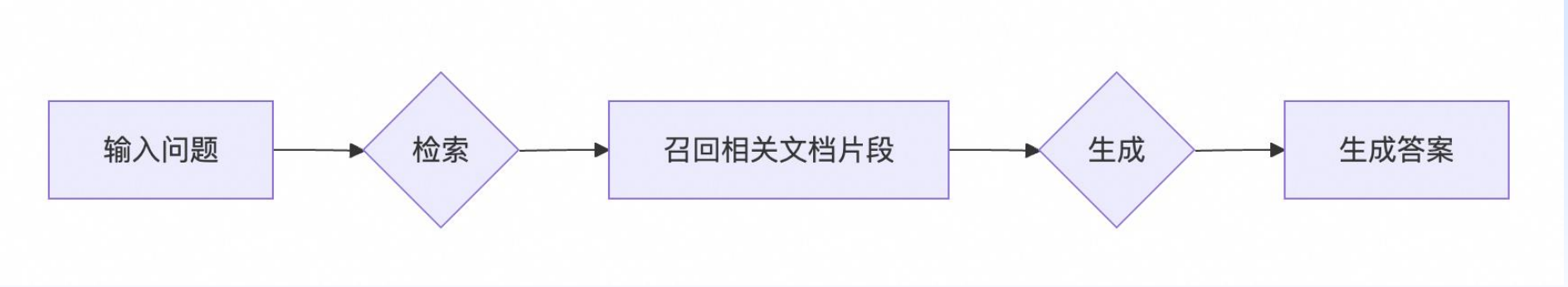
为什么用了RAG，我的AI还是笨得跟猪一样？

digoal 德哥

目录

RAG是什么

- RAG（Retrieval-Augmented Generation）
  - 检索增强生成
  - 将检索外部知识和大模型生成结合，提升AI对新知识的理解和生成能力
- 例子
  - 假如你问AI：“蜜柑这部小说讲了什么故事？”
    - RAG会先检索包含“蜜柑”的相关文档片段，再结合这些信息生成回答。



为什么要用RAG

- 大模型训练数据有限，无法覆盖所有领域和最新内容
- 训练大模型昂贵，企业/个人难以持续更新
- 很多企业/行业数据无法用于训练（如医疗、内部手册等）
- RAG 可以让大模型临时“学会”知识，扩展其能力
- 例子
  - 医疗、法律、公司知识库等私有数据，无法直接训练进大模型，通过RAG可以在问答时临时引用。

为什么用了RAG还是效果不好

- 常见问题举例
- 例子
  - 问：张伟是哪个部门的？
  - 召回内容没覆盖相关知识如“教研部” → AI无法正确回答
  - 召回信息太杂乱→ AI被误导，答非所问

回答正确性	召回覆盖率	召回精度	主要原因
低	高	高	模型本身能力不足
低	高	低	召回了太多无关信息，噪声影响生成
低	低	高	召回条数不够，没有涵盖答案所有观点
低	低	低	解析、切分、向量化、召回全流程都存在问题

如何评判RAG的效果

- 主要评测维度

指标	说明	计算方式/例子
回答准确率	答案与标准答案的语义和事实准确性	F1分数、语义相似度+事实准确度加权
召回覆盖率	召回内容对答案涉及观点的覆盖比例	context_recall = 匹配数 / 总观点数
召回精度	召回内容相关性及排名准确性	context_precision，考虑排名和相关程度

示例

question	answer	ground_truth	contexts	context_recall	context_precision
0 张伟是哪个部门的?	根据提供的信息，没有提到张伟所在的部门。如果您能提供更多关于张伟的信息，我可能能够帮助您找到答案。	张伟是教研部的成员	["提供行政管理与协调支持，优化行政工作流程。"，"绩效管理部 韩杉 李飞 I902 041 ..."]	0.0	0.0
1 张伟是哪个部门的?	张伟是人事部门的	张伟是教研部的成员	["李凯 教研部主任"，"牛顿发现了万有引力"]	0.0	0.0
2 张伟是哪个部门的?	张伟是教研部的	张伟是教研部的成员	["牛顿发现了万有引力"，"张伟 教研部工程师，他最近在负责课程研发"]	**1.0**	**0.5**

召回覆盖率和召回精度得分算法

在Ragas 中，context recall 用来描述 ground_truth 中有多少比例的观点可以得到 contexts 的支持，计算过程如下：

- 由大模型将 ground_truth 分解成 n 个观点（statements）。
  - 比如，可以由ground_truth「张伟是教研部的成员」生成观点列表 [张伟是教研部的]。
- 由大模型判断每个观点能在检索到的参考资料（contexts）中找到依据，或者说 context 是否能支撑 ground_truth 的观点。
  - 比如，这个观点在第三行数据的 contexts 中能找到依据「张伟 教研部工程师，他最近在负责课程研发」。
- 然后 ground_truth 观点列表中，能在 contexts 中找到依据的观点占比，作为 context_recall 分数。
  - 这里的得分为 `1 = 1/1`。 `contexts中匹配了几个观点 / ground_truth有几个观点`

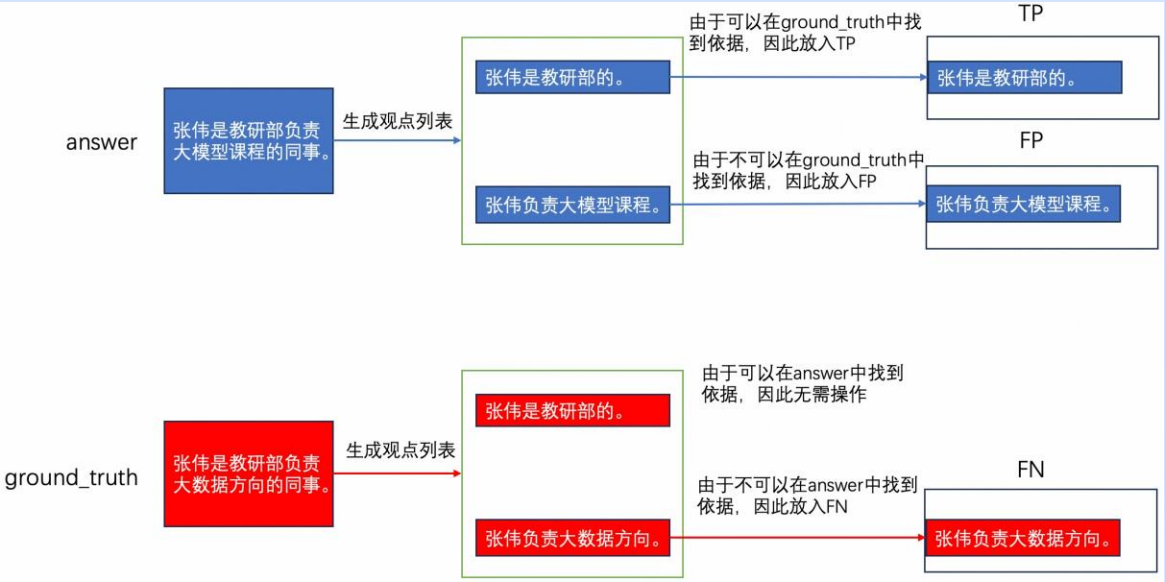
`Context precision`：召回文本与问题和正确答案的相关性, 以及越相关的文本是不是排在越前面? 有价值的数据排在越前面, 分就越高.

在Ragas 中，context precision 不仅衡量了 contexts 中有多少比例的 context 与 ground_truth 相关，还衡量了 contexts 中 context 的排名情况。计算过程比较复杂：

- 按顺序读取 contexts 中的 contexti ，根据 question 与 ground_truth，判断 contexti 是否相关。相关为 1 分，否则为 0 分。
  - 比如上面第三行数据中，context1(牛顿发现了万有引力) 是不相关的，context2 相关。
- 对于每一个 context，以该 context 及之前 context 的分数之和作为分子，context 所处排位作为分母，计算 precision 分。
  - 每个context算出一个得分: `1或0/其位置`
  - 对于上面第三行数据，context1 precision 分为 `0/1 = 0`，context2 precision 分为 `1/2 = 0.5`。
- 对每一个 context 的 precision 分求和，除以相关的 context 个数，得到 context_precision。
  - 对于上面第三行数据， `context_precision = (0 + 0.5) / 1 = 0.5`。

answer得分算法

- 把正确答案拆分成若干个观点
- 把RAG的回复结果也拆分成若干观点
- 遍历answer与ground_truth列表，并初始化三个列表，TP、FP与FN



Answer
FULL OUTER JOIN
Truth

对于由answer生成的观点：

- 如果该观点与ground_truth的观点相匹配，则将该观点添加到TP列表中。比如：「张伟是教研部的」。
- 如果该观点在 ground_truth 的观点列表中找不到依据，则将该观点添加到FP列表中。比如：「张伟负责大模型课程」。

对于ground_truth生成的观点：

- 如果该观点在 answer 的观点列表中找不到依据，则将该陈述添加到FN列表中。比如：「张伟负责大数据方向」。
- 该步骤的判断过程均由大模型提供。

answer得分算法

1.2.4、统计TP、FP与FN列表的元素个数，并按照以下方式计算f1 score分数：

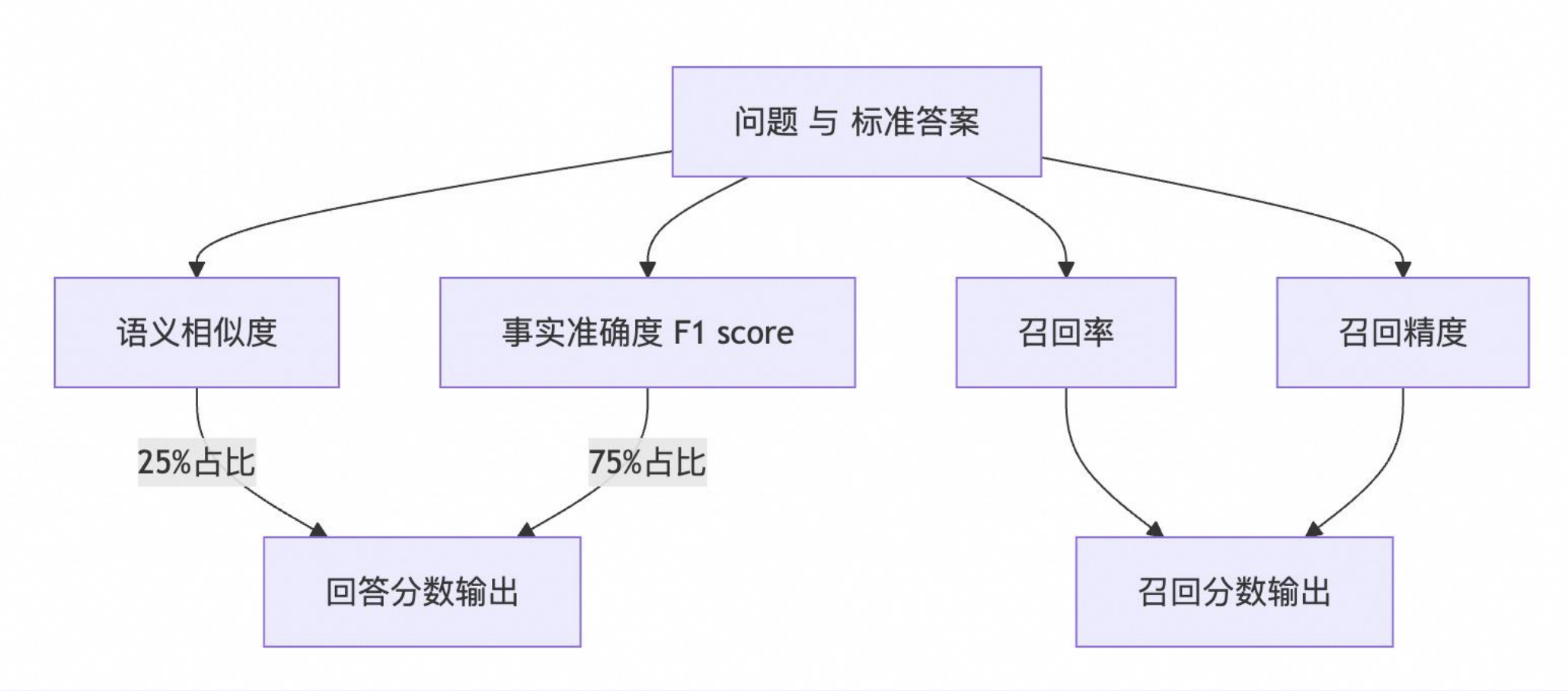
```
f1 score = tp / (tp + 0.5 * (fp + fn)) if tp > 0 else 0
```

以上文为例：`f1 score = 1/(1+0.5*(1+1)) = 0.5`

1.3、分数汇总, 得到语义相似度和事实准确度的分数后，对两者加权求和，即可得到最终的 Answer Correctness 的分数。

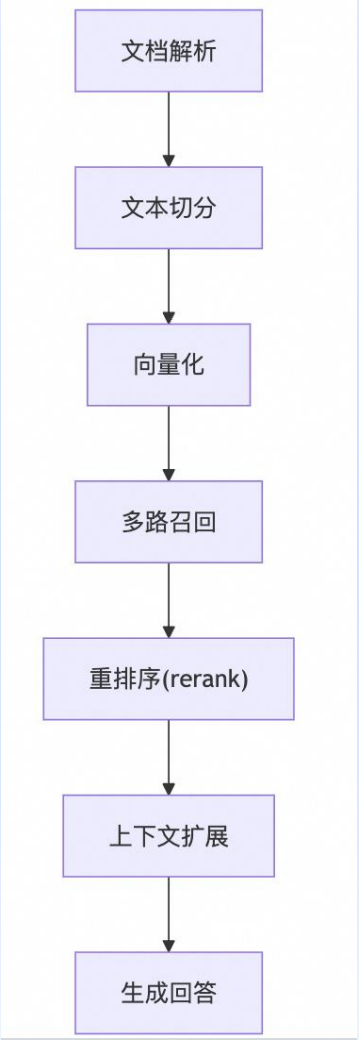
```
Answer Correctness 的得分 = 0.25 * 语义相似度得分 + 0.75 * 事实准确度得分
```

Ragas评测流程简图



RAG的整体优化思路

- 目标:
- 提高
  - 召回覆盖率
  - 召回精度



- 文档解析：格式统一，内容清洗
- 切分方法：按tokens/句/段/章节/语义等
- 向量化：选用新embedding模型
- 多路召回：语义向量、分词、模糊、标签、BM25等
- Rerank：用模型或bm25提升相关性排序
- 上下文扩展：补齐上下文片段，防止信息断裂
- 生成：优选大模型

优化建议举例：

- 召回无关信息多→引入rerank
- 召回条数太少→提升topN
- embedding模型老旧→升级模型
- 切分粒度不合适→调整切分参数

基于维基百科数据的RAG优化实践

测试用到的数据库: PolarDB for PostgreSQL

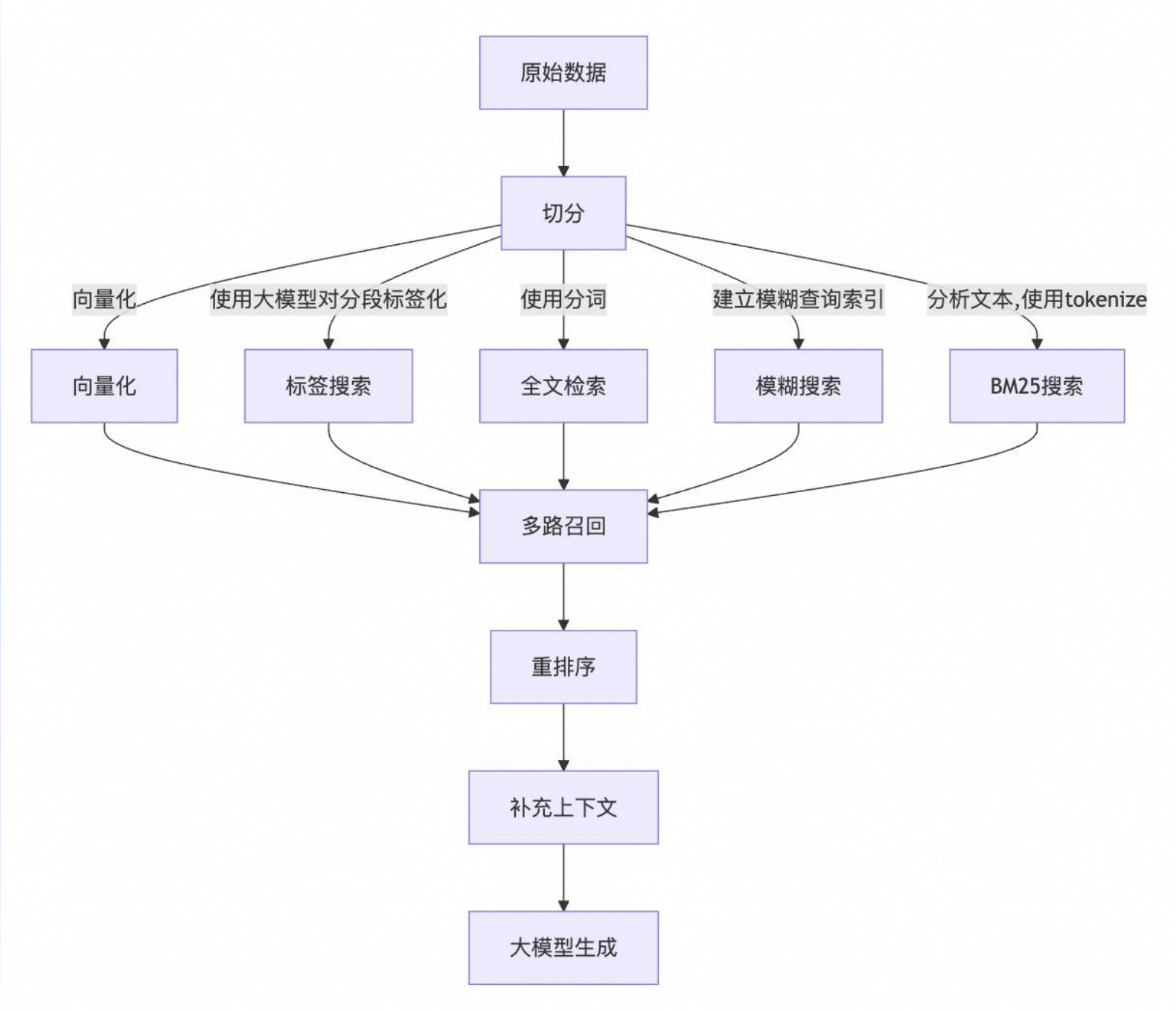
- 环境: 《穷鬼玩PolarDB RAC一写多读集群系列 | 在Docker容器中用loop设备模拟共享存储搭建PolarDB RAC》

使用到的插件:

- pgml
- openai
- http
- pg_jieba
- pg_bigm
- vector
- pg_tokenizer
- VectorChord-BM25

PolarDB 公众号

流程


原始数据
切分
向量化
使用大模型对分段标签化
使用分词
建立模糊查询索引
分析文本,使用tokenize
向量化
标签搜索
全文检索
模糊搜索
BM25搜索
多路召回
重排序
补充上下文
大模型生成

流程

1. **数据准备**
   - 下载“wikimedia/wikipedia”中文数据集
   - DuckDB/Pg导入，字段：“id, url, title, text”
2. **分段切分**
   - 使用pgml(chunk)等分段函数
   - 切分参数如chunk_size=500, overlap=100
   - 分段方法:

```
SPLITTERS = {
    "character": CharacterTextSplitter,
    "latex": LatexTextSplitter,
    "markdown": MarkdownTextSplitter,
    "nltk": NLTKTextSplitter,
    "python": PythonCodeTextSplitter,
    "recursive_character": RecursiveCharacterTextSplitter,
    "spacy": SpacyTextSplitter,
}
```

3. **多路召回之, 标签 & QA提取 & 全文检索 & 模糊查询**
   - 用大模型总结每段关键Q/A、关键词标签，存入表
4. **多路召回之, 向量化与索引**
   - embedding模型：如mxbai-embed-large
   - 建立向量、分词、模糊、标签倒排索引
5. **rerank**
   - 语义topN、分词topN、标签topN、BM25topN等
   - 合并后用rerank模型或BM25再排序，选topK
6. **上下文扩展+RAG生成**
   - 召回片段前后N段补齐上下文
   - 问题+召回内容输入大模型生成最终回答

BM25

- BM25是经典的文本相关性打分算法
- 兼顾词频（TF）和逆文档频率（IDF），适合排序召回结果
- 全局有意义：常用词权重低、稀有词权重高
- 适合中文等多语言（配合分词）
- 可作为多路召回的重要一环，提升相关性排序
- 例子
  - “蜜柑”是问题关键词，BM25能根据其在所有文档的稀有度和在每段的出现频率精准打分

使用bm25例子:

```
-- create a tokenizer
SELECT create_tokenizer('bert', $$
model = "bert_base_uncased"  # using pre-trained model
$$);

-- tokenize text with bert tokenizer
SELECT tokenize('A quick brown fox jumps over the lazy dog.', 'bert')::bm25vector;

-- Output: {1012:1, 1037:1, 1996:1, 2058:1, 2829:1, 3899:1, 4248:1, 4419:1, 13971:1, 14523:1}
-- The output is a bm25vector, 1012:1 means the word with id 1012 appears once in the text.
```

```
CREATE TABLE documents (
    id SERIAL PRIMARY KEY,
    passage TEXT,
    embedding bm25vector
);

-- create a text analyzer which uses jieba pre-tokenizer
SELECT create_text_analyzer('text_analyzer1', $$
[pre_tokenizer.jieba]
$$);

SELECT create_custom_model_tokenizer_and_trigger(
    tokenizer_name => 'tokenizer_jieba1',
    model_name => 'model_jieba1',
    text_analyzer_name => 'text_analyzer1',
    table_name => 'documents',
    source_column => 'passage',
    target_column => 'embedding'
);
```

```
- 在 bm25vector 列上创建索引，以便我们可以收集全局文档频率。
CREATE INDEX documents_embedding_bm25 ON documents USING bm25 (embedding bm25_ops);


- 现在我们可以计算查询和向量之间的 BM25 得分。
- 需要注意的是，bm25 得分为负数，这意味着得分越高，文档的相关性就越高。
- 我们特意将其设为负数，以便您可以使用默认的 order by 来优先获取最相关的文档。

SELECT id, passage, embedding <&> to_bm25query('documents_embedding_bm25', tokenize('人', 'tokenizer1'))
FROM documents
ORDER BY rank
LIMIT 10;

 id |                                                                                                  passage
----+-----------------------------------------------------------------------------------------------------------------------------------------------------
 11 | 这女人的漂亮丈夫，在旁顾而乐之，因为他几天来，香烟、啤酒、柠檬水沾光了不少。
  9 | 那几个新派到安南或中国租界当警察的法国人，正围了那年轻善撒娇的犹太女人在调情。
  1 | 红海早过了，船在印度洋面上开驶着，但是太阳依然不饶人地迟落早起，侵占去大部分的夜。
```

总结

- RAG让AI拥有“**临时学习**”能力，对接企业私有知识库，解决大模型“幻觉”与知识盲区
- 评测RAG效果必须**量化**：召回覆盖率、召回精度、回答准确率
- 优化RAG是一项**系统工程**：数据解析、切分、标签、向量化、检索、重排序、上下文扩充、生成都要关注
- **BM25**等传统检索算法在混合召回和排序阶段依然不可或缺
- 实践证明，严谨的流程+细致的优化+多路召回+再排序，才能让RAG“聪明起来”
- 更多发展关注: KAG、GraphsRAG等论文

谢谢

- 参考文稿:
- https://github.com/digoal/blog
  - 《为什么用了RAG, 我的AI还是笨得跟猪一样! RAG效果评测与优化》
  - 《维基百科(wikipedia) RAG 优化 | PolarDB + AI》
  - 《召回精度提升插件: pg_tokenizer + VectorChord-BM25 reranking》
  - 《AI论文解读 | KAG》
- 阿里云大模型专家ACP认证 学习课程

digoal 微信

digoal 公众号

PolarDB 公众号