搜索系统改造提议（草案）

Memory-Search Project

2026 年 1 月 16 日

目录

1 背景与现状

当前 `/search` 接口的实现流程如下：

1. 在 `search_views` 表上使用 PostgreSQL Full-Text Search（FTS）召回：
   - 使用 `sv.fts @@ plainto_tsquery('simple', :q)` 做匹配
   - 使用 `ts_rank_cd` 计算相关性分数
   - 使用 `ts_headline` 生成高亮片段
2. 通过 `JOIN problem_items pi ON pi.id = sv.item_id` 补全题目实体信息：
   - 标题：`COALESCE(NULLIF(pi.problem_text,''), pi.bm25_text)`
   - 标签、图片、元数据过滤
3. Python 层按 `item_id` 聚合多行视图：
   - 题目分数 = 所有视图行中最大 rank
   - 每题保留 top3 evidence
   - 最终返回 top_n 个题目

该方案实现简单、易于调试，适合项目早期验证。

2 当前主要问题

2.1 LIMIT 是行级而不是 item 级

SQL 中的 `LIMIT :k` 作用在 `search_views` 的“行”上，而不是“题目”上。如果同一题目在多个 `view_type` 中都命中，会占用多个名额，导致：

- 召回结果的题目多样性下降
- 少数强题垄断召回池

2.2 单一路召回容易偏科

当前系统只依赖一条召回通道（FTS + 原始 query）：

- 用户描述偏“解析/方法”，但题干窗口更强
- query 包含年份、地区、别名等需要归一化
- 有些需求更像模糊匹配而非全文检索

2.3 不同窗口分数不可比

未来一旦引入多窗口、多 query rewrite，不同窗口的 `ts_rank_cd` 分布差异很大，直接加权求和会导致尺度灾难。

3 改造目标

- 召回更稳定：对不同描述方式都有覆盖。
- 结果可解释：evidence 能体现命中来源窗口/字段。
- 改动可控：复用现有表结构与接口，渐进升级。
- 便于调参：可通过权重/窗口开关做 A/B。

4 方案：多窗口召回 + 融合重排

4.1 窗口（Window）的定义

窗口 = 一条召回通道（query 版本 × 视图范围 × 检索方式）。
初始建议窗口：

- W1：题干窗口（`problem_text, diagram_desc`）
- W2：解析窗口（`solution_outline, method_chain`）
- W3：记忆窗口（`user_notes, user_tags`）
- W4：短 query 窗口（关键词抽取后的 query）

4.2 窗口内输出：item 级 topM

每个窗口执行一次 SQL，输出：

- `item_id`
- `window_score`（如窗口内最大 rank）
- 窗口内 topK evidence
- 标题、标签、图片

原则：每个窗口返回的是“题目级”topM，而不是“行级”topM。

4.3 融合策略：RRF（名次融合）

由于不同窗口分数不可比，第一版推荐使用 RRF：
若某 item 在窗口 i 的名次为 r_i，则该窗口贡献为：

$$score_i = \frac{1}{k + r_i}$$

最终融合分：

$$FinalScore = \sum_i weight_i \cdot \frac{1}{k + r_i}$$

其中 k 常取 60，$weight_i$ 为窗口权重。
优点：

- 不依赖原始分数尺度
- 多窗口共同命中会自然靠前
- 实战稳定、参数少

4.4 第二阶段：归一化 + 加权和

在积累用户日志后，可升级为：

- 每窗口分数做归一化（分位数、z-score）
- 再加权求和或使用学习排序

5 数据与接口设计

5.1 接口保持兼容

继续返回：

- `item_id, title, score, evidence, user_tags, images`

建议 evidence 中新增可选字段：

- `window`：来源窗口
- `raw_rank`：行级 rank
- `window_score`：窗口内聚合分

5.2 窗口配置化

窗口可用 Python 配置：

- 窗口名
- 允许的 `view_type`
- query 变体
- 权重

6 实施路线

6.1 Phase 0：稳定现状

- 保证现有接口可跑通
- 补齐建表、索引、刷新 `search_views` 的脚本

6.2 Phase 1：多窗口，多 SQL

- 多跑几次 SQL（不同窗口）
- Python 中按 item 聚合
- 用 RRF 融合

6.3 Phase 2：窗口内聚合下推到 SQL

- SQL 中 `GROUP BY item_id`
- 输出窗口内 item 级分数

6.4 Phase 3：日志与学习

- 增加点击、停留、收藏等埋点
- 调窗口权重或学习排序

7 评估指标

7.1 离线指标

- Recall@K
- MRR / nDCG
- 覆盖率（题干型、记忆型、解析型 query）

7.2 在线指标

- 点击率 CTR@K
- 首点位置
- query 改写次数
- 无结果率

8 风险与注意事项

- 第一版避免对原始 rank 直接加权
- 控制窗口数量与每窗 topM
- 最终 evidence 需跨窗口筛选最有解释力的
- meta_filters 建议作为 hard filter

9 附：当前代码跑通的最短 Checklist

1. 数据库中存在 `problem_items` 与 `search_views`
2. `search_views` 含：`item_id, view_type, fts, text`
3. `fts` 与查询均使用 `'simple'` 字典
4. 对 `fts` 建 GIN 索引
5. 在 psql 中手动验证 FTS 查询
6. 再启动 FastAPI 调用 `/search`