# 电网知识自适应学习系统项目方案汇报

# 一、项目背景

当前基于电网知识的学习系统存在一些亟需解决的问题：

1. 重复题目过多：现有题库系统通常按固定顺序或随机出题，缺少对题目语义相似度的分析，导致用户在短期内重复练习同类题目，浪费学习时间。

2. 复习不精准：传统复习机制往往基于固定间隔或固定题库顺序，未能考虑用户对不同知识点和难度的掌握情况，难以实现个性化复习。

3. 学习效率低：由于重复题目和非个性化复习的存在，用户需要更多时间才能达到同等学习效果，难以高效掌握知识点。

此外，用户能力差异也影响学习效率：

1. 能力水平不均：不同用户对题目的理解和掌握能力差异较大，如果系统未能根据能力调整题目难度，会出现题目过难或过易的情况，降低学习效果和用户体验。
2. 难度适配不足：传统系统无法动态评估用户能力并调整题目难度，导致“简单题浪费时间、复杂题挫败感高”，不利于持续学习。
3. 复习调度缺乏个性化：同一知识点对不同用户的重要性和掌握程度不同，但系统缺乏动态调度策略，无法优化每个用户的复习路径。

因此，为了提升电网知识学习系统的效率与个性化体验，本项目引入自适应学习机制。通过结合题目语义分析、用户能力动态评估以及智能复习调度，系统可以：

* 减少重复练习：基于题目语义相似度自动识别相似题，降低重复练习概率。
* 个性化难度匹配：根据用户实时能力值动态选择题目，使题目难度既不过易也不过难，提升学习效率与成就感。
* 智能复习调度：结合间隔复习策略与用户掌握情况，优化复习顺序和间隔，确保关键知识点得到充分巩固。
* 数据驱动优化：记录用户作答行为和知识点掌握情况，为持续优化学习策略和复习计划提供科学依据。

这一机制不仅改善了现有系统的低效问题，还能为用户提供更精准、灵活和可持续的学习体验，为电网知识学习提供智能化支持。

# **二、项目目标**

本项目旨在构建一个自适应学习系统，通过题目难度与用户能力的动态匹配、相似题降频机制以及智能复习调度，实现高效知识掌握与持续复习。具体目标包括：

1. 相似题降频/跳过

* 目标：在用户做对复杂题后，自动减少或延迟相似简单题的出现，避免重复练习和时间浪费。
* 实现机制：通过题目语义向量化和近邻检索，系统可识别题目之间的相似性，并结合题目难度及用户能力值动态调整出现频率。
* 价值：提升学习效率，使用户将更多时间集中在尚未掌握或高价值的题目上，减少低价值重复练习。

1. 可配置复习频率

* 目标：用户可根据自身学习节奏和目标，自主设置复习强度，支持预设档位（轻度、标准、强化）以及完全自定义模式。
* 实现机制：系统采用间隔复习模型（Leitner/SM2 简化版），结合用户作答结果动态调整复习间隔和优先顺序。
* 价值：增强学习的个性化体验，让复习既不过于频繁造成负担，也不会因为间隔过长而遗忘知识点，实现科学化、可持续的知识巩固。

1. 记录与统计

* 目标：全面记录用户作答历史、能力曲线、复习进度及知识点掌握情况，为学习效果分析和策略优化提供数据基础。
* 实现机制：系统对每题作答结果、时间、正确率及复习状态进行记录，并支持按知识点、难度、时间维度进行统计与可视化分析。
* 价值：为用户提供清晰的学习反馈和成长曲线，帮助教师或管理员了解整体学习情况，同时为算法优化提供数据支持，实现闭环改进。

综上，本项目通过相似题降频、可配置复习频率和全面记录统计三大核心功能，实现了对用户学习过程的智能化管理。相似题降频机制有效减少重复练习，提高学习效率；可配置复习频率保证了复习的个性化和科学性，使知识点得到合理巩固；全面的记录与统计功能不仅为用户提供清晰的学习反馈和成长曲线，也为管理员及系统优化提供数据支持。整体来看，这三者协同作用，为电网知识学习构建了一个高效、智能且可持续的自适应学习体系。

# **三、数据处理与向量索引**

## 3.1 文本向量化策略

为了实现题目语义的精准表示，系统对题目文本进行统一的拼接与向量化处理，参与向量化的字段包括：question（题干）、options（选项，如有）、doc、standard、field、knowledge\_points。其中，为避免泄露答案：answer 和 analysis 不参与向量化，以防止“答案语义”影响相似度计算，避免相似题匹配偏差。

字段清洗与拼接：question 支持 str/list/dict 三种形式，列表或字典先转换为可读字符串。options 若为列表按自然顺序拼接。

拼接格式按段落显式分隔：

[Question]\n{question}

[Options]\n{A)\nB)\n...}

[Doc]\n{doc}

[Standard]\n{standard}

[Field]\n{field}

[Knowledge]\n{k1; k2; ...}

此策略增强不同属性的可区分性，同时严格避免答案/解析混入向量。

## **3.2 难度映射**

为了实现题目难度与用户能力的动态匹配，系统对原始题目难度进行了统一整数映射，为后续打分、筛选和策略计算提供标准化基础：

统一映射规则：历史题库中难度通常用 L1~L3 表示，系统将其映射为整数 1~3，即：L1→1、L2→2、L3→3。统一映射可使算法在处理不同题库或历史数据时保持一致性，便于打分函数计算、能力匹配和复习优先级排序。

健壮性处理：对于未知、缺失或异常值，系统默认映射为 L1（最简单），保证算法稳定性，避免异常值导致打分或复习调度出错。在 UI 层仍尊重用户对难度的选择和筛选，保证前端展示与用户偏好一致。

自适应逻辑中的应用：系统将 L3 题目视为“复杂题”，用于触发相似题抑制策略。即当用户完成复杂题后，系统会根据题目语义相似度和难度差异，自动减少或延迟相近简单题的出现，从而避免低价值重复练习。难度映射也用于能力适配：打分函数会根据用户当前能力值与题目难度的差值调整适配度权重，使题目难度既不过易也不过难，提升学习效果。

对统计与分析的支撑：映射后的整数难度可用于统计分析和趋势追踪，如按难度分布计算正确率、曝光次数和复习覆盖率。便于生成能力曲线和复习报告，为教师、管理员及系统优化提供科学依据。

扩展灵活性：未来可在映射基础上增加子层级（如 1.0~3.0 连续值），进一步提高难度适配精度。难度映射与知识点掌握状态结合，可实现更精细的个性化复习策略。

## **3.3 Qdrant 数据访问与向量检索机制**

### 3.3.1 Qdrant 数据访问与约定

#### 3.3.1.1 连接与集合

在本系统中，题目向量数据存储和检索依赖于 Qdrant 向量数据库。Qdrant 内置 HNSW（Hierarchical Navigable Small World）索引结构，结合 cosine 相似度作为距离度量，实现高效的近邻搜索。这一设计可以保证在海量题库下，仍能快速响应相似题目检索请求。

为了提高系统性能，客户端采用单例模式或惰性初始化策略，并默认启用 gRPC 协议（prefer\_grpc=True）。这样可以避免每次请求都重新初始化连接，降低网络和系统开销，同时保证客户端调用的稳定性。

集合（Collection）命名为 power\_QA\_Benchmark，用于存储题目向量及题目元信息（metadata）。在集合中，每条数据既包含题目向量本身，也保存了题目的来源、类型、难度、知识点等信息，以支持丰富的查询、过滤和评分功能。

#### 3.3.1.2 Point ID与Payload

1. Point ID 使用策略：

如果题目原始 ID 为 uint64 类型，系统直接使用该 ID，确保数据一致性。对于非 uint64 类型或字符串形式的题目 ID，系统生成 UUIDv5 来保证 gRPC 调用的唯一性和兼容性，同时避免 ID 冲突。

1. Payload 内容及用途：

每个向量点对应的 payload 包含以下元信息：source\_id：原始题目 ID，用于题目关联与查询；doc：题目所属文档或标准；field：题目所属领域或模块；type：题目类型（如选择题、判断题等）；difficulty：题目难度等级；standard：题目遵循的行业标准；knowledge\_points：题目涉及的知识点标签。

这些元信息的存在，使得系统不仅可以进行向量相似性检索，还能实现基于属性的过滤。例如，应用端可以筛选指定难度或指定知识点的题目，提高推荐的精准性。同时，这些元信息也可用于 UI 展示和打分模块（Scorer）逻辑计算。

### 3.3.2 向量生成与存储

系统将题目内容（question + options + doc + standard + field + knowledge\_points）整合后生成向量，不包含答案与解析信息。每道题生成一个 1024 维浮点向量，直接写入 Qdrant 集合中进行持久化存储。原始题目 ID 保存在 payload.source\_id 中，便于后续检索和关联操作。通过这种方式，系统可以保证向量与题目本身一一对应，同时满足在线检索和离线分析需求。

### 3.3.3 向量可用性检测

在实际运行中，题库中可能存在部分题目尚未生成向量，或向量存储失败的情况。为了保证在线检索的安全性，系统提供 vector\_exists 功能，用于判断题目是否已入库并拥有有效向量。

功能说明

1). 安全检索保障：在执行基于向量的相似题检索之前，通过 vector\_exists 判断向量是否存在，避免因缺失向量导致检索报错或结果异常。

2). 降级处理：对于部分向量化的数据集，即使部分题目缺失向量，也能正常执行其他评分或推荐逻辑，不会影响整体系统运行。

实现逻辑：

1). 通过 point\_id 检查：如果题目已知 point\_id，系统优先调用 Qdrant 的 retrieve 或 get\_points 接口判断向量是否存在。

2). 通过 source\_id 检查：若仅有题目 source\_id，则使用 scroll + filter 方法查询 payload 中的 source\_id 字段，判断向量状态。

3). 缓存优化：系统对热点题目启用 LRU 缓存，避免重复的数据库访问。缓存命中时，直接返回向量存在性结果，进一步降低延迟。

返回值

1). 布尔值：True 表示题目向量存在，False 表示不存在。

2). 可选 point\_id：在必要时，返回对应 point\_id 供上层模块复用，避免重复计算或查询，提高整体效率。

通过上述机制，vector\_exists 功能保证了系统在大规模题库环境下的稳定性和可扩展性，同时支持在线检索和离线分析的高效协作。

# **四、向量与相似度在自适应调度中的应用**

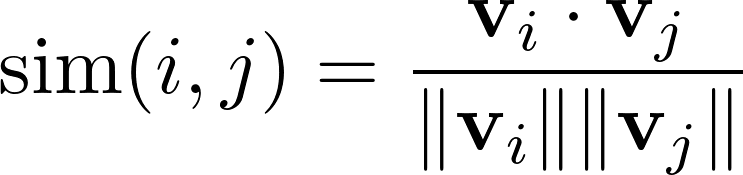
## **4.1 向量生成与相似度计算**

在自适应学习系统中，实现题目语义分析是核心环节之一。通过将题目内容转化为向量表示，可以在数学空间中衡量题目间的语义相似度，从而为相似题识别、重复题抑制、错题强化以及个性化题目调度提供可靠依据。

**向量来源：**系统对每道题目的关键字段进行拼接，形成文本表示：

T=concat(question, options, doc, standard, field, knowledge\_points)

**相似度计算：题目向量之间的相似度使用 余弦相似度：**



在 Qdrant 向量数据库中，集合 power\_QA\_Benchmark 使用 cosine 距离度量，其优化目标等价于最大化上述相似度。

通过该计算方式可以精准识别语义相似但表述不同的题目，为在线近邻检索提供基础，实现快速准确的相似题查找，从而支持复杂题抑制简单题、错题强化相似题、新颖度控制等策略，有效避免低价值重复练习，显著提升学习效率。

## **4.2 在线最近邻检索（Qdrant）**

在题目调度环节，需要快速获取“源题”的语义最接近题目集合，用于相似题抑制、错题强化及个性化抽样。

检索接口设计可参考下述伪代码，其中：1.filters用于映射 UI 选项，例如 field/type/difficulty/standard。未选即表示全选；2.返回 top-K 近邻及相似度分数，用于后续打分逻辑。

def search\_neighbors\_by\_id(point\_id, top\_k, filters):

q\_filter = build\_qdrant\_filter(filters)

return qdrant.search(

collection="power\_QA\_Benchmark",

query={"point\_id": point\_id},

limit=top\_k,

filter=q\_filter,

)

向量存在性检测中优先通过缓存命中查询 point\_id，未命中则通过 payload.source\_id 执行 scroll 查询。为确保在线检索安全，即使部分题目未向量化，也不会影响系统运行。

def vector\_exists(source\_id):

return cached\_or\_lookup\_point\_id(source\_id) is not None

返回结构统一屏蔽内部 point\_id，使系统模块解耦且便于前端或打分模块使用。

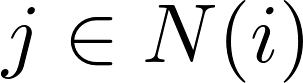
[{ "id": source\_id, "score": sim, "payload": {...} }, ...]

## **4.3 最近邻在打分器中的应用（抑制策略）**

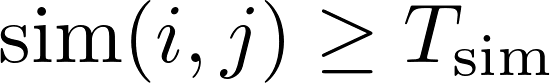
最近邻算法在打分器中依然发挥了很大的作用，结合用户最近作答历史，通过近邻关系调整候选题目分数，实现“相似题抑制、错题强化、新颖度控制”，增强个性化和学习效率。最终目标为：减少重复、低价值的简单题出现，提高学习效率。

**复杂题正确后的相似简单题抑制：**

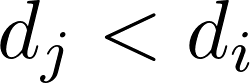
条件1: 候选题为 i 的邻居



条件2:



条件3: 候选题难度低于复杂题



温和模式罚分公式：



严格模式可直接剔除 j 若干轮。目的是：减少重复、低价值的简单题出现，提高学习效率。

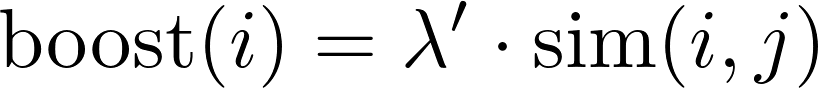
for i in recent\_correct\_complex:

for (j, sim) in neighbors(i):

if sim >= tau\_sim and diff(j) < diff(i):

score[j] -= lambda\_suppress \* sim \* (diff(i) - diff(j) + 1)

错题的相似题会进行强化，围绕错误题目强化变式题训练，提升弱项掌握。计算公式为：

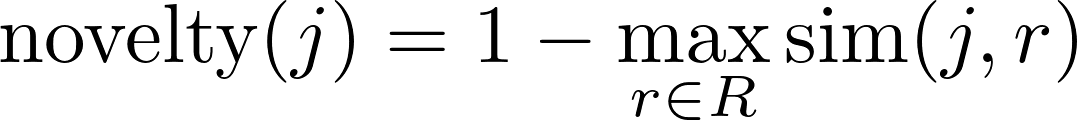


for i in recent\_wrong:

for (j, sim) in neighbors(i):

score[j] += lambda\_boost \* sim

另一方面，对新颖度也需要进行控制，增加题目多样性，降低重复感，同时也可以避免重复练习，有效提升学习效率。



## **4.4 综合打分与题目抽样**

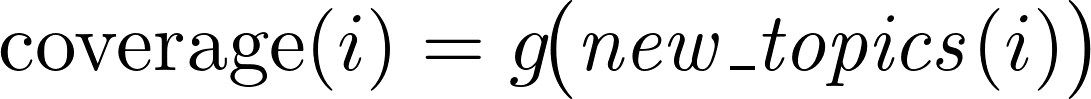
能力-难度匹配：用户能力为ɑ，题目难度为d



复习时，根据复习到期/逾期时间加权：



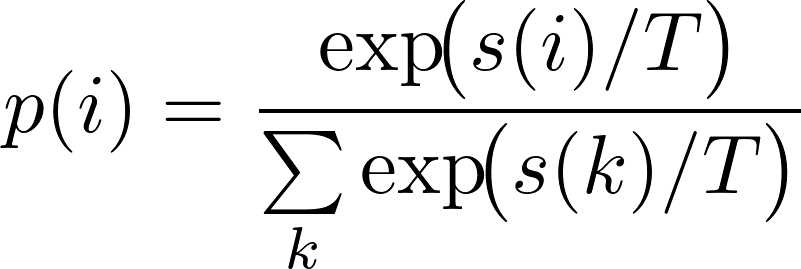
同时覆盖未复习的知识点：



总分计算的公式为：



若抑制/剔除导致题量不足，从原候选顺序补齐，保证本轮题量稳定。softmax 采样（温度 τ）为：



## **4.5 筛选一致性与容错机制**

在所有邻居检索环节统一嵌入 UI 过滤器，确保“抑制/强化”策略仅在用户圈定的题目空间内生效，避免跨域干扰；若向量缺失或检索失败导致邻居为空，相似度相关项自动跳过，其余打分逻辑无缝执行，保证推荐不中断；同时对高频组合 (i, filters, K) 的结果实施短期缓存，显著降低 Qdrant 查询压力，提升接口响应速度。

## **4.6 参数建议与调优**

| **参数** | **建议值** | **说明** |
| --- | --- | --- |
| τsim | 0.6~0.75 | 相似度阈值 |
| λ,λ′ | 3~10 | 抑制/强化强度 |
| softmax τ | ≈0.5 | 题目抽样温度 |
| 用户能力初值 a | 1.0 | 题目难度 L1~L3 对应整数 1~3 |
| 正确/错误步长 | ±0.15 | 能力动态调整步长 |

# **五、复习调度机制设计与实现**

## **5.1 核心目标**

复习调度模块的核心目标，是在学习过程中合理控制每道题目何时再次出现，确保用户对已学知识能够按照科学的节奏进行复习，从而形成长期稳定的记忆路径。传统题库系统通常按照固定顺序或随机出题，这种方式无法区分哪些题已经熟练掌握、哪些题需要重复巩固，容易导致重复练习或遗忘过快。

为此，本系统引入自适应复习调度，主要实现以下目标：

1. 按节奏回访：根据题目掌握情况和复习历史，合理安排题目的出现时间，既避免过早重复，也防止知识点遗忘。
2. 逾期优先：对已到期或逾期的题目，赋予更高优先级，让用户更快回顾关键知识点。
3. 与整体自适应协同：复习调度与能力匹配、相似题抑制/强化、新颖度控制等机制无缝配合，实现全局自适应的学习路径。

通过这一机制，用户不仅可以获得按需复习的个性化体验，还能在长期学习过程中保持高效记忆与稳定进度。

## **5.2 题目状态模型**

为了实现复习节奏控制，系统为每道题维护一组轻量级状态信息，这些状态用于记录题目的掌握情况和复习时间节点：

| **属性** | **类型** | **说明** |
| --- | --- | --- |
| bucket | int (0..4) | Leitner 层级编号，层级越高，复习间隔越长，表示题目掌握程度 |
| next\_ts | float | 下一次应该复习的时间戳（秒） |
| last\_ts | float | 最近一次作答时间戳 |
| last\_result | bool | 最近一次作答是否正确 |
| correct\_streak | int（可选） | 连续答对次数，可用于微调间隔策略或奖励连续正确题目 |

状态模型设计的原则是轻量、可解释、易持久化。即使在大规模题库环境下，也能快速更新状态并支持持久化保存。

## **5.3 间隔策略设计**

复习间隔是控制题目出现节奏的核心参数。系统允许用户根据个人学习节奏和目标选择不同强度的复习策略：

| **强度** | **默认间隔（天）** |
| --- | --- |
| 轻度 | [1, 3, 7, 21] |
| 标准 | [0.5, 2, 5, 14] |
| 强化 | [0.25, 1, 3, 7] |
| 自定义 | 用户可输入 4 级间隔数组 |

在实际代码中，天数会转换为秒数进行计算。

对于答错题目，还会设置 短间隔短回看（short\_interval\_wrong，目前设置为10分钟或1小时），确保用户能快速再次尝试，避免知识遗忘或理解偏差过久未纠正。

通过这种灵活配置，系统可以兼顾不同用户的学习节奏需求，从轻度回顾到强化记忆都能适配。

## **5.4 答题事件驱动的状态更新**

复习调度遵循 简化版 Leitner/SM2 策略，通过题目作答事件驱动状态更新。具体逻辑如下：

1. 正确作答：题目掌握程度提高，bucket向上调整，复习间隔延长。

bucket = min(bucket + 1, 4)

next\_ts = 当前时间 + intervals[bucket]

1. 错误作答：题目掌握程度下降，bucket向下调整，并设置短间隔快速回看。

bucket = max(bucket − 1, 0)

next\_ts = 当前时间 + short\_interval\_wrong

伪代码示意：

def on\_answer(item, is\_correct, now):

b = item.bucket or 0

if is\_correct:

b = min(b + 1, 4)

item.next\_ts = now + intervals[b]

else:

b = max(b - 1, 0)

item.next\_ts = now + short\_interval\_wrong

item.bucket = b

item.last\_ts = now

item.last\_result = is\_correct

## **5.5 复习优先分计算**

为了在选题时体现题目的复习紧迫性，系统设计了 复习优先打分 s\_rev，用于给到期或逾期题目加分，使其在候选题排序中靠前。

计算逻辑如下：

1. 只有到期或逾期才加分：若当前时间早于 next\_ts，则 s\_rev = 0。
2. 逾期越久，加分越多：

* Δ = max(0, now−next\_ts)（逾期时长）
* base = 当前 bucket 对应的间隔
* s\_rev = w\_rev·log(1+Δ/ base)，对极端逾期做对数压缩，避免分数异常放大。

1. 可选线性简化版：

s\_rev = w\_rev·min(1,Δ/ base)

伪代码：

def review\_score(item, now):

if item.next\_ts is None or now < item.next\_ts:

return 0.0

delta = now - item.next\_ts

base = intervals[max(item.bucket or 0, 0)]

return clip(w\_rev \* math.log1p(delta / base), 0.0, s\_rev\_max)

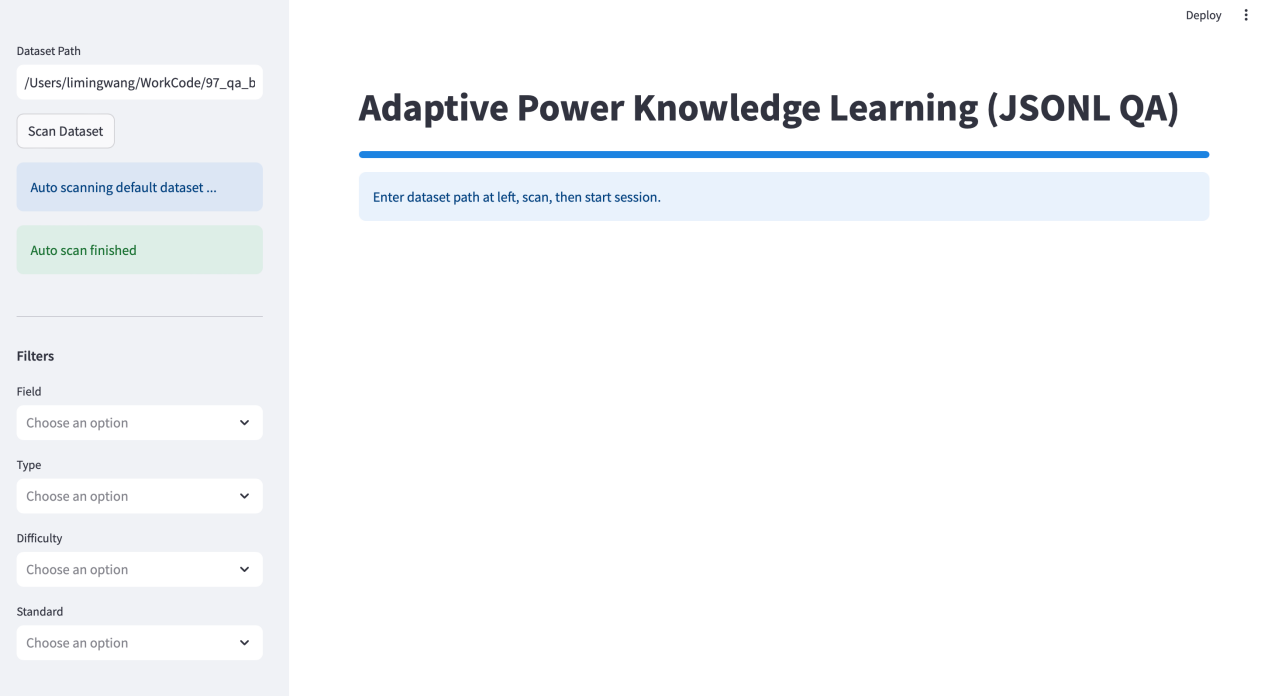
## **5.6 参数建议与调优**

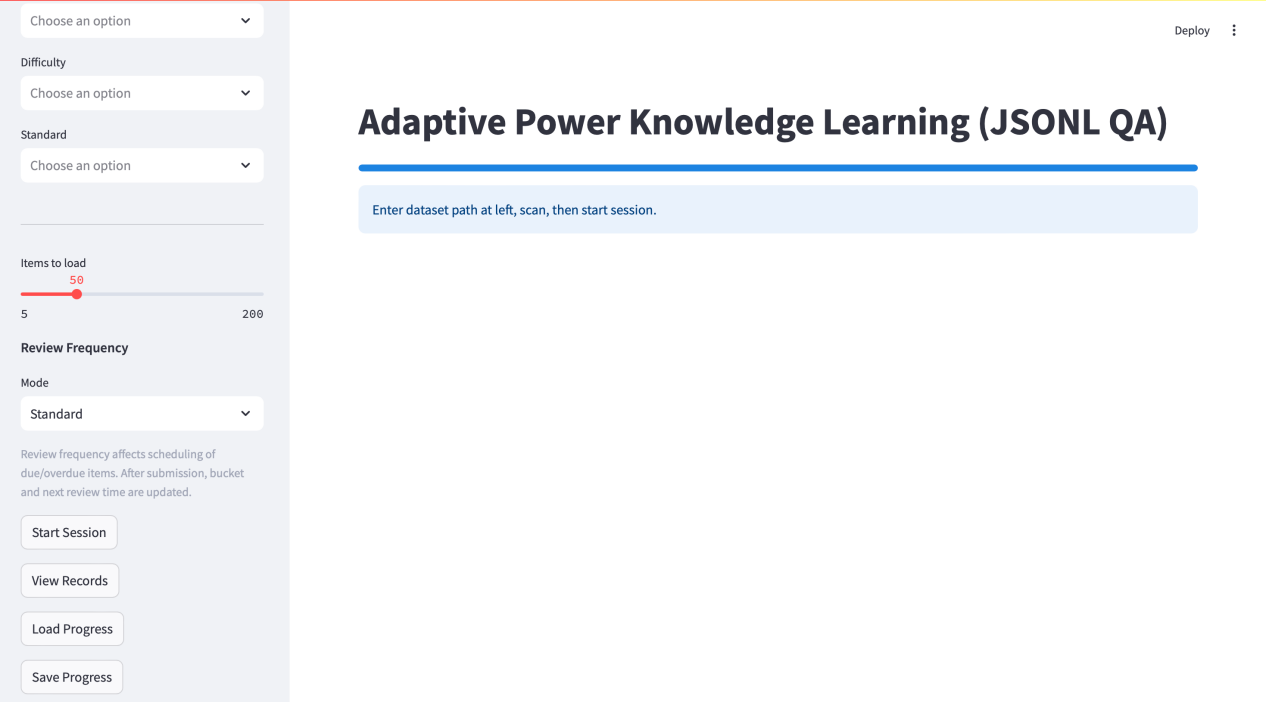
| **参数** | **建议值** | **说明** |
| --- | --- | --- |
| intervals（标准） | [0.5,2,5,14] 天 | 4级复习间隔 |
| short\_interval\_wrong | 10 分钟 | 错题短回看间隔 |
| w\_rev | 1.0 | 复习优先权重 |
| s\_rev\_max | 2.0 | 复习优先分上限 |

# **应用页面概览**

## 6.1 应用主页面与左侧工具栏

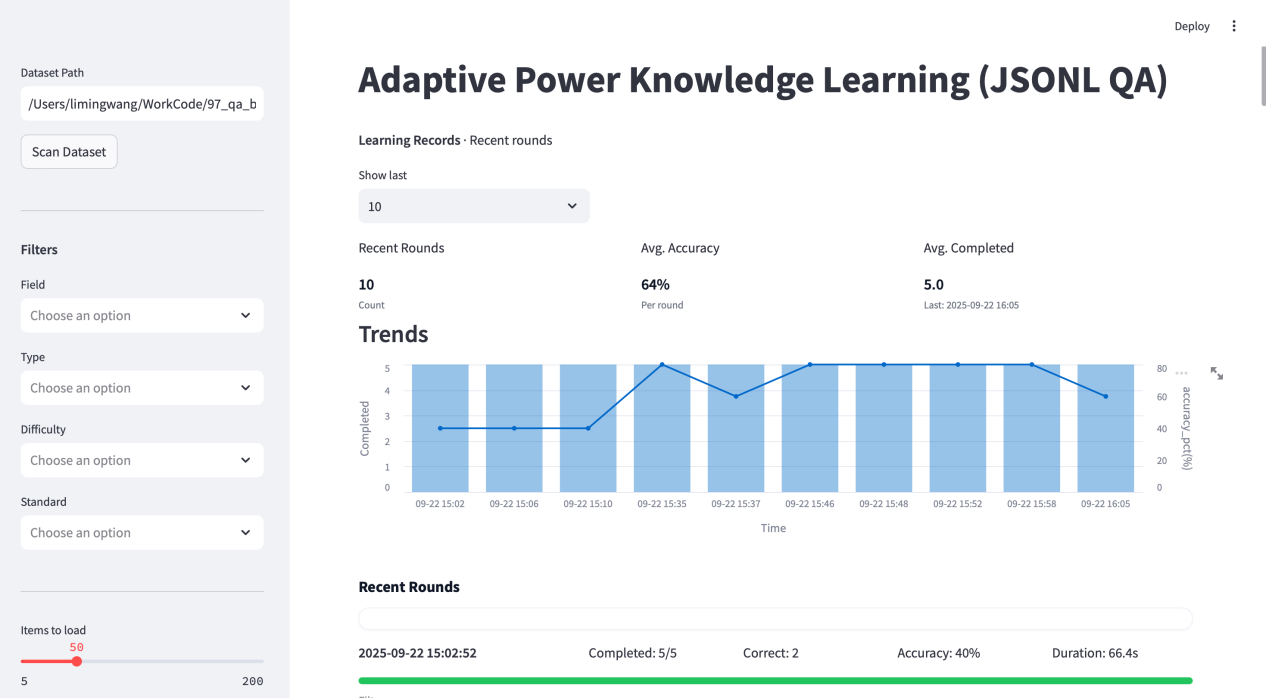
主页面可以对题库的数据集进行选择。在工具栏侧主要功能为筛选题目，包括领域/类型/难度（L1–L3）等；如果不选，默认视为“全选”。还可以设置题量与复习频率及查看学习记录，包括进入历史页（趋势、最近轮次、明细）等。

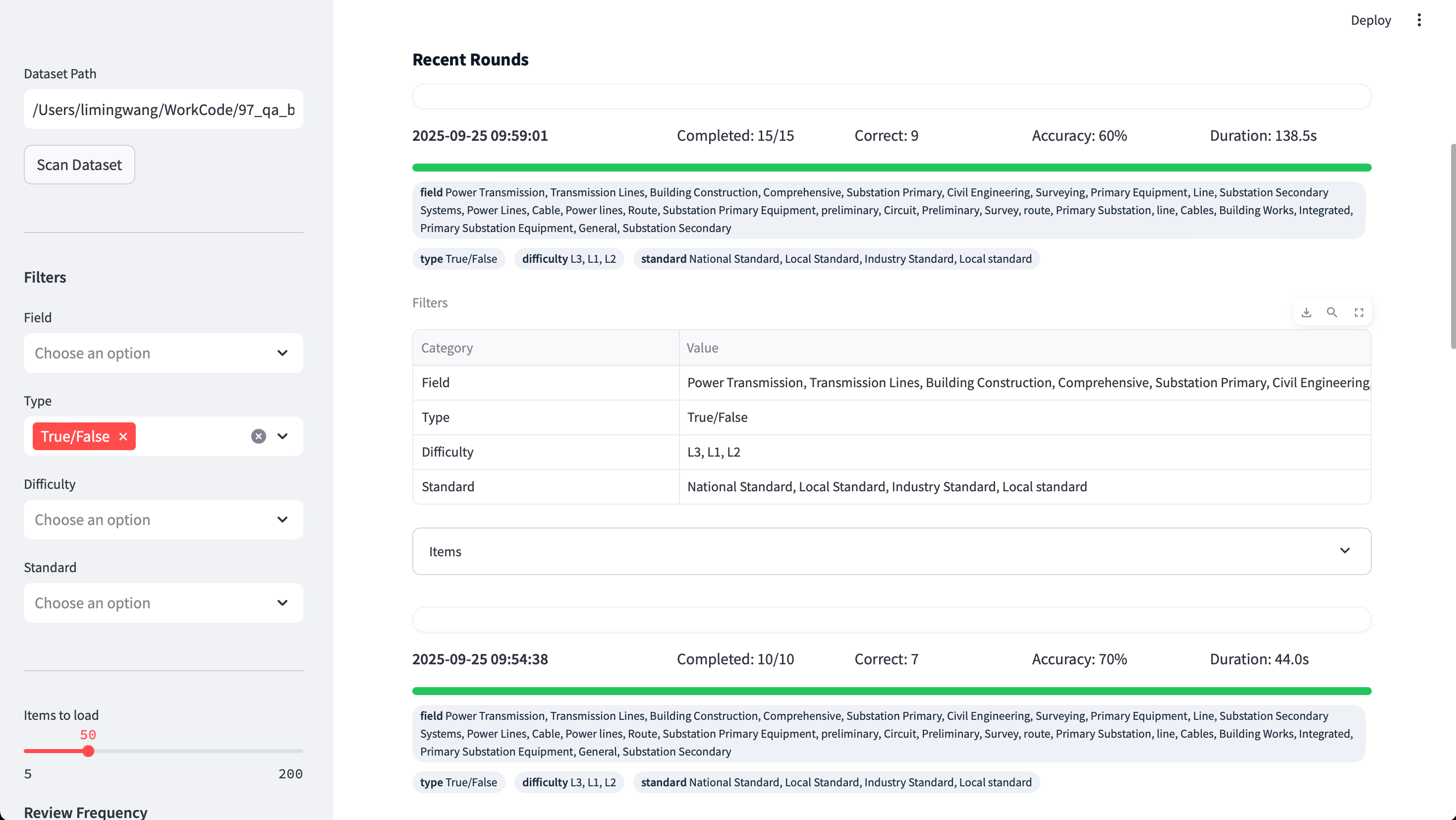


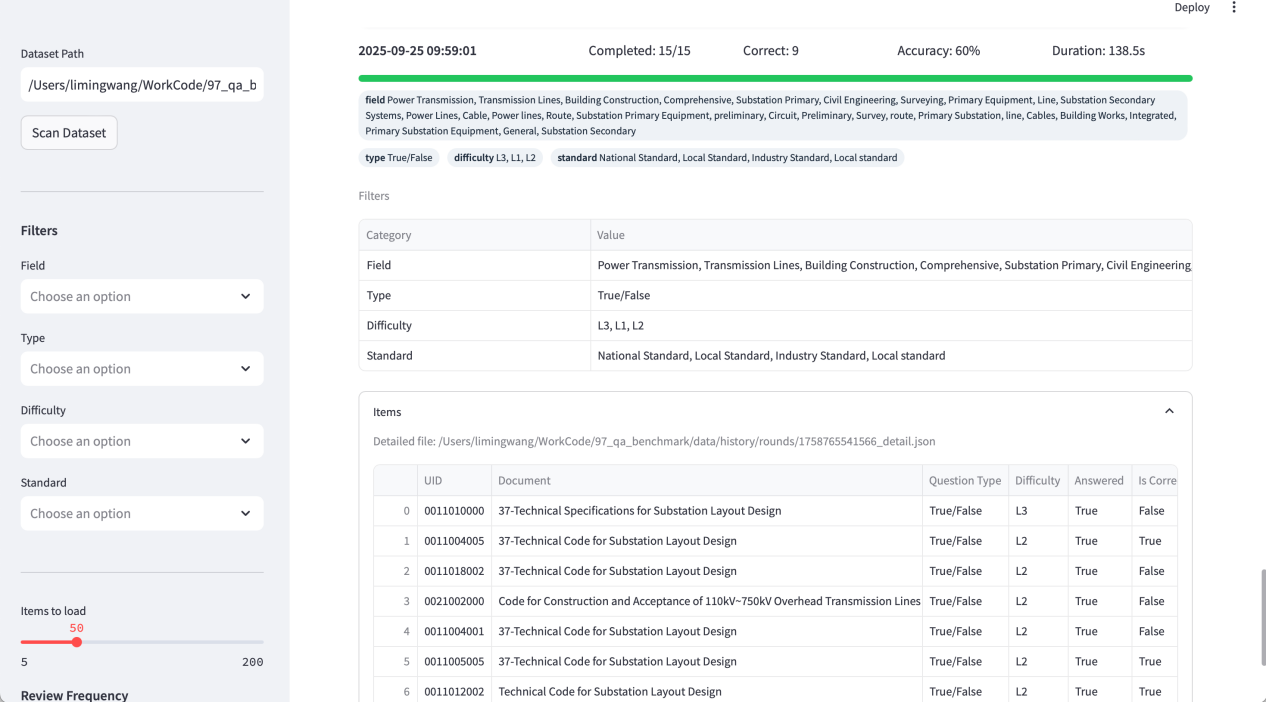


## 6.2 学习记录功能界面

在学习记录功能界面中，可以清楚看到历史学习记录的趋势图（时间、题目完成数量与准确率）、最近轮次及所做题目的明细数据（题目内容，是否正确等）。







## 6.3 答题界面与本轮统计

顶部显示已答数量、正确数量、正确率；能力值随对/错即时±0.15 调整。只有全部题目作答后，“提交并判分”才进入总结；否则提示未完成，不自动跳转。汇总页面中包含难度分布、概览本轮覆盖的领域/知识点/文档等。

