北京大学深圳研究生院
Peking University Shenzhen Graduate School

AgentEvolver: Towards Efficient Self-Evolving Agent System

Project: https://arxiv.org/abs/2511.10395
Code: https://github.com/modelscope/AgentEvolver

汪静雅

2025.12.04

01 Abstruct

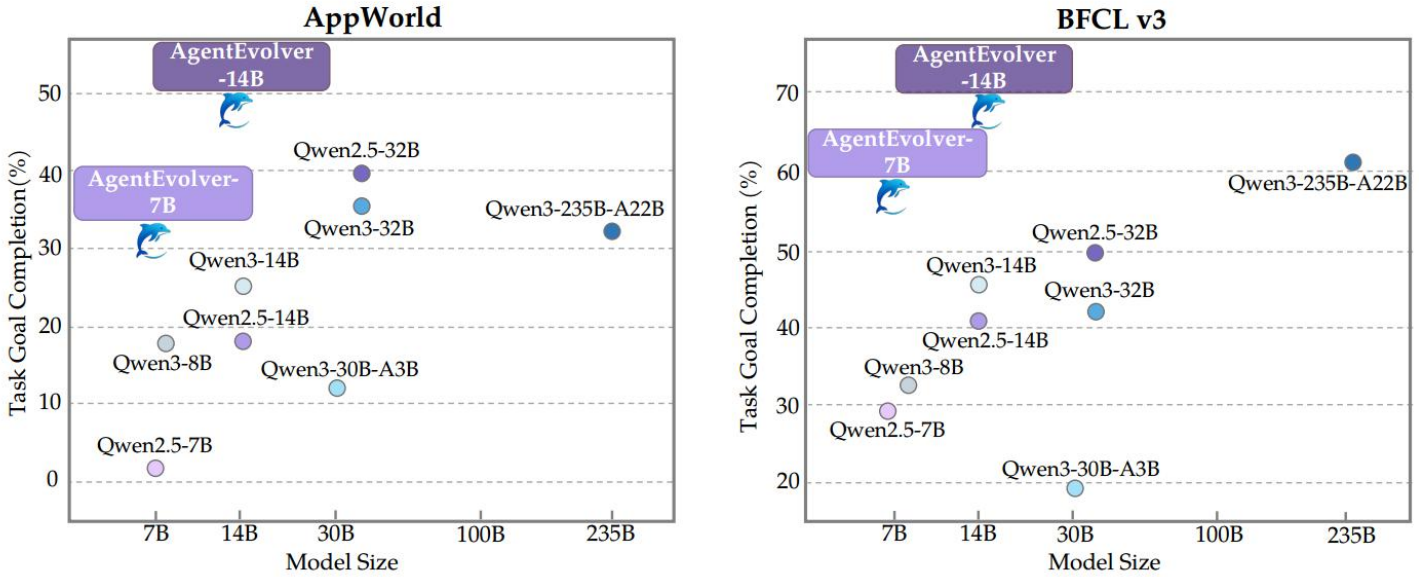
研究背景: 大语言模型驱动的智能体，能通过推理、工具使用完成复杂任务，有望提升人类生产力。但当前智能体开发依赖人工任务数据集 + 强化学习（RL），存在数据构建成本极高；探索效率低下；样本利用率差的三大问题。

研究重点:

- **提出自进化智能体系统，利用 LLM 的语义理解与推理能力，实现自主学习，核心创新是 3 大协同机制:**
- 自提问（Self-Questioning）；自导航（Self-Navigating）；自归因（Self-Attributing）

研究成果:

- 与传统的基于RL的基线相比，AgentEvolver实现了更高效的探索、更好的样本利用率和更快的适应。



目前研究现状：

智能体开发的两大范式

- **工作流编排范式**：灵活性差
- **基于学习**：
  - 任务构建成本高-新环境中工具功能未知，人工设计轨迹耗时
  - 探索效率低-长任务中随机探索冗余轨迹多
  - 样本利用率差-依赖海量轨迹采样，近似暴力搜索

工作架构：

自提问：实现好奇心驱动的探索，使LLM能够通过探索环境的状态-动作空间和发现功能边界来自主生成任务。这种机制减少了对昂贵的手工数据集的依赖。

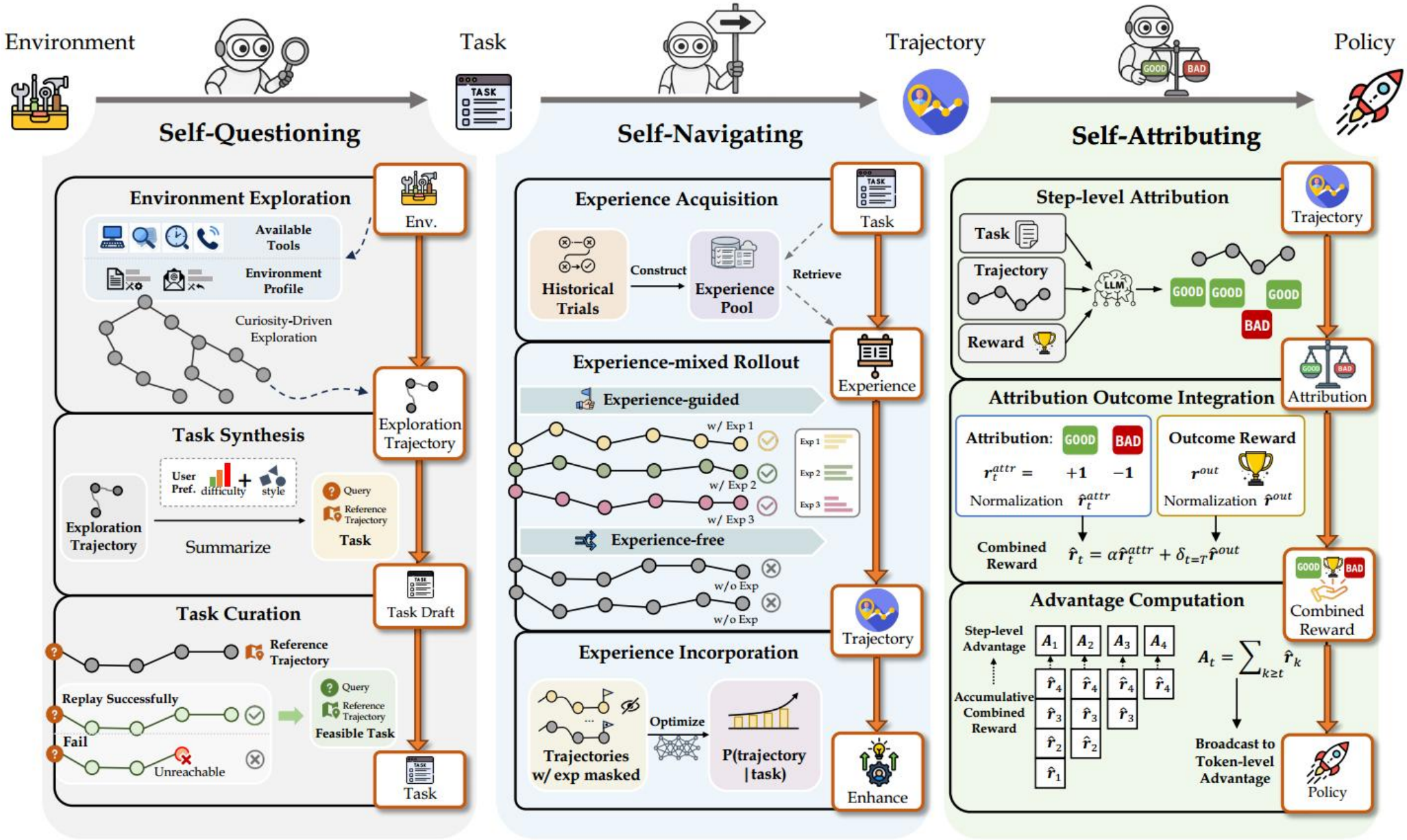
↓

自导航：通过重用和概括过去的经验来提高探索效率。通过混合策略学习和轨迹引导，代理实现了更有针对性和更高效的任务完成。

↓

自归因：通过沿长轨迹分配细粒度奖励来提高样本效率。LLM不是像典型的GRPO方法那样统一地归因结果，而是推断中间状态和动作的各自贡献，从而实现更细粒度和有效的学习。

北京大学 信息工程学院
PEKING UNIVERSITY School of Electronic and Computer Engineering



i）从环境到任务
ii）从任务到轨迹
iii）从轨迹到策略

核心真实目标：

让智能体通过与环境$\mathcal{E}$交互，通过自主探索来估计$p_{target}(g)$，进而自主生成任务、奖励信号，实现策略提升，而不依赖人工预设任务或奖励。

- **交互沙盒（环境）：** $\mathcal{E} = (\mathcal{S}, \mathcal{A}, \mathcal{P})$，其中 $\mathcal{S}$：状态空间；$\mathcal{A}$：动作空间；$\mathcal{P}$：状态转移概率

它定义了环境状态与动作，但无预设奖励函数 / 任务目标，是 “开放无奖励” 环境 。

- **目标任务空间和奖励：** $p_{target}(g)$ 代表智能体最终必须掌握的“黄金”任务。

每个任务g对应一个期望的目标状态s_g，存在一个真实奖励函数$R_g(s,a)$

因此，智能体的真实目标即学习一个优化策略$\pi_\theta(a|s,g)$，最大化目标任务的期望回报 $J_{target}(\theta)$

$$J_{\text{target}}(\theta) = \mathbb{E}_{g\sim p_{\text{target}},\, s_0\sim p_0}\left[V^{\pi_\theta}(s_0,g)\right],$$

$$V^{\pi_\theta}(s_0,g) = \mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t R_g(s_t,a_t)\,\middle|\, s_0,g,\pi_\theta\right],$$

自进化的代理目标：

由于目标任务 / 奖励未知，智能体需自主构建代理任务 + 代理奖励：

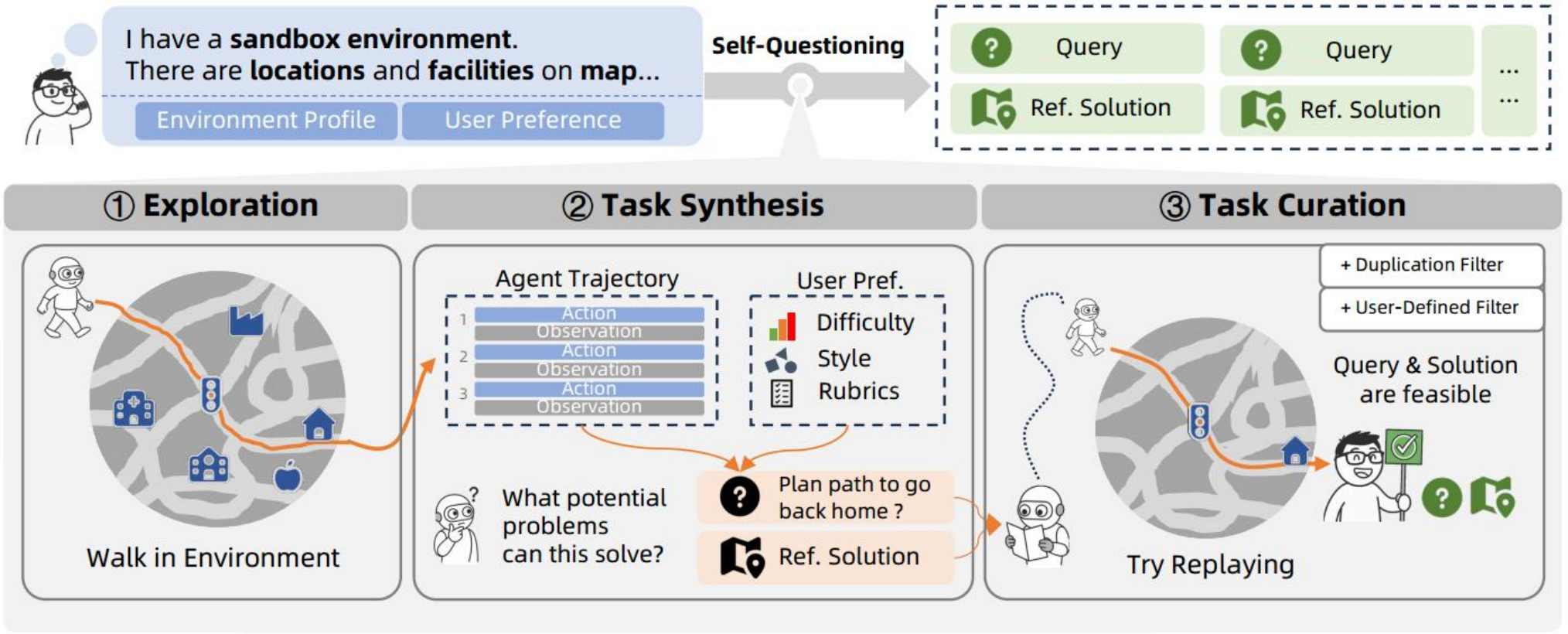
- **代理任务生成：**代理通过自我提问机制探索环境，以生成一组候选的可解决任务，实现从环境到任务的映射：$F_{\text{task}} : \mathcal{E} \to \Delta(\mathcal{G}), \quad \text{where} \quad p_{\text{train}} = F_{\text{task}}(\mathcal{E})$
  p_{train}是具有适当难度和多样性的任务，模拟未知的目标p_{target}
- **代理奖励设计：**代理通过自归因推断奖励函数，实现从环境和任务到奖励函数的映射：

$$F_{\text{reward}} : \mathcal{E} \times \mathcal{G} \to (\mathcal{S} \times \mathcal{A} \to \mathbb{R}), \quad \text{where} \quad \hat{R}_g = F_{\text{reward}}(\mathcal{E}, g)$$

这实现了比稀疏环境奖励更细粒度的信用分配，使智能体能够在没有人为干预的情况下从更丰富的反馈中学习。

自我进化的核心目标是制定和优化 F_{task}和F_{reward}函数，以最大化训练目标 $J_{train}(\theta)$；即使得在“代理任务与代理奖励”下优化得到的策略π_θ，能够在真实的目标任务分布上表现良好。

04 Self-Questioning



自我提问模块试图回答数据生成中的四个主要问题：

1. 如何理解一个未知的环境，并在其中找到有价值的状态？
2. 如何生成符合用户偏好的任务，同时保持足够的多样性以满足异构需求？
3. 如何避免幻觉，保证任务的可执行性？
4. 如何提供代理奖励信号，以评估代理在轨迹级别的表现？

1. 如何理解一个未知的环境，并在其中找到有价值的状态？-带环境配置文件的好奇驱动探索

- **Environment Profiles as Action Priors**：为了有效地获得处理复杂环境的相关能力，通过**环境配置文件**捕获这些规律，这些配置文件总结了实体、属性和支持的操作。

An example of *Environment Profile*

```
Entity: Map
a game map with navigable roads and several points of interest, including a hospital,
school, factory, home (or houses), and traffic lights.
Attributes:
- road: a traversable pathway.
- architecture: a structure (e.g., hospital, school, home, or factory).
- traffic_light: a signal requiring agents to wait before crossing.
Operation:
- move: move to an adjacent location.
- wait_and_cross: wait for the signal to change and cross the intersection.
- enter: enter the nearest building.
```

- **LLM-Driven Stochastic Exploration**：
  - **高温LLM生成动作**：高温促使智能体发现非传统的相互作用模式和隐藏的环境状态，从而产生多样化和创造性的行为。
  - **两阶段探索平衡效率**：
  - 初始N_b步：广度优先探索，建立对环境基础语义的初步理解，确保覆盖动作 - 状态空间的多样区域；
  - 后续步骤：深度优先探索，聚焦此前发现的有潜力轨迹；在该阶段，通过 “短视决策规则” 防止过早收敛到单一行为模式，即智能体仅考虑最近N_d步的观测

这一步实现了从环境$\mathcal{E}$到轨迹分布的转化。这一过程由 LLM 驱动的随机策略$\pi_{explore}$引导，形式化表示为：

$$\Phi : \mathcal{E} \times \pi \times \mathcal{S} \to \mathcal{T}, \quad \text{where} \quad \rho = \Phi(\mathcal{E}, \pi_{\text{explore}}, s_0),$$

ρ是探索得到的轨迹分布

2. 如何生成符合用户偏好的任务，同时保持足够的多样性以满足异构需求？-自适应任务合成

- **偏好引导的任务合成：**任务难度和任务风格（任务难度由潜在解决方案中涉及的实体、属性和操作的数量来量化。任务风格由用户定义的量规来指定，确保任务反映用户需求）

$$\Psi : \mathcal{T} \times \mathcal{U} \to \mathcal{G}, \quad \text{where} \quad g = \Psi(\rho, u) \sim p_{\text{train}},$$

- **参考解提取：**任务合成的关键是提供可信的参考解，而任务是探索后生成 的，因此解已包含在前期探索轨迹中；将探索轨迹简化为 “动作 - 观测” 格式；把简化轨迹和对应任务输入 LLM，提取完整步骤，得到参考解。

这一步生成了难度可控、风格匹配用户需求的任务，同时附带可靠参考解，实现轨迹向任务的映射。

3. 如何避免幻觉，保证任务的可执行性？-任务筛选与分布混合

- **任务筛选：**1. 生成中过滤，词汇重叠去重（ token 相似度超过阈值则丢弃）与语义相似度检查；2. 生成后过滤，先重复词汇去重；再用任务的参考解在环境中实际执行，验证是否可完成，保证可行性

- **分布混合（可选）：**可根据情况加入一部分真实数据作为参考，平衡稳定性与性能扩展。

$$p_{\text{train}}(g) = F_{\text{task}}(\mathcal{E})(g). \quad p_{\text{hybrid}}(g) = (1-\lambda)p_{\text{target}}(g) + \lambda p_{\text{task}}(g),$$

4.如何提供代理奖励信号，以评估代理在轨迹级别的表现？-基于 LLM 判断的合成任务奖励

- **LLM as judge**：作为F_{reward}的后备代理，在没有特定优化但具有代理能力的情况下，跨不同环境提供通用评估。LLM遵循两个打分原则：
  - **相关性和重复性检查**：作为基础筛选，识别与任务无关的步骤、幻觉 / 冗余步骤，出现此类行为的智能体直接得 0 分。
  - **连续评分**：提供细粒度反馈。正确完成任务得高分，错误尝试得低分，分数还会结合 “步骤效率、中间错误” 等因素调整，捕捉部分进度以辅助智能体提升。

- **参考解辅助的正确性检查**：为保证奖励客观性，LLM 裁判会结合探索阶段得到的参考解对比智能体轨迹与参考解，验证是否覆盖关键步骤（包括直接匹配或功能等价的替代步骤）。

这一步产生了客观的奖励标准。

05 Self-Navigating

复杂环境中，传统 RL 依赖随机试错，导致轨迹冗余、收敛慢；而人类会从过往经验中学习并迁移知识。因此提出**自导航机制**，通过 “经验复用 + 轨迹引导 + 混合策略学习”，将探索从无引导试错转向结构化自提升。

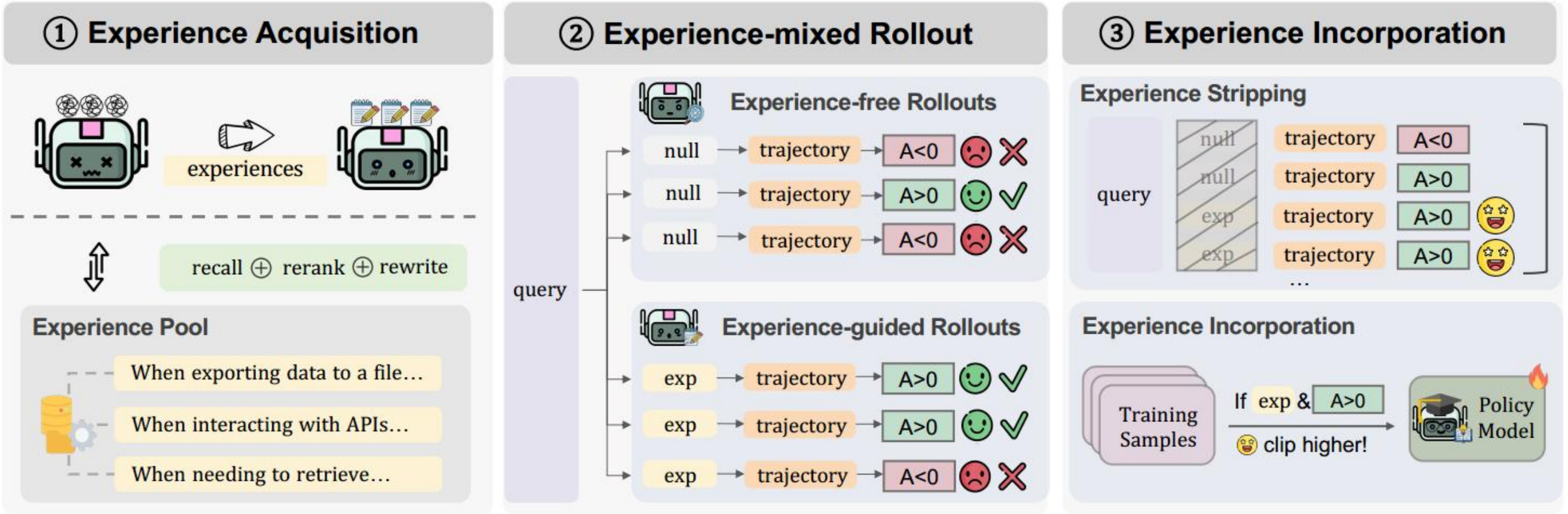
- **核心概念experience：** “经验” 是从过往轨迹中提炼的结构化知识（包含成功 / 失败案例），用于引导智能体决策，形式为自然语言描述，包含 2 个核心组件：
  - **When to use：**适用场景（触发条件），作为检索依据（通过语义匹配其向量嵌入）；
  - **Content：**具体指导（指令、注意事项、策略），用于在对应场景中辅助动作选择。

  （示例：在 AppWorld 中，“当使用未确认的 API 时，先通过文档验证其存在 / 功能”）

An example of *experience* for AppWorld

When to use:
When attempting to use an API that hasn't been explicitly confirmed to exist or function as expected.

Content:
Always verify the existence and behavior of an API through its specifications (using apis.api_docs.show_api_doc) before executing calls, especially for critical actions like deletions or modifications.



(i) **经验获取**：智能体从过往轨迹中提炼出结构化的自然语言经验，并对其进行构建与检索；
(ii) **经验混合轨迹生成**：将 “无引导轨迹” 与 “经验引导轨迹” 交织结合，以平衡探索与利用；
(iii) **经验融合**：将检索到的经验整合到策略优化过程中。

1. Experience Acquisition

为了实现有效的经验重用，我们引入了一个跨越离线和在线阶段的经验获取阶段。

- **池构建（离线阶段）**：对P_{train}中每个任务g，用初始策略生成N_{pc}组独立轨迹；每条轨迹轨迹经 4 步处理：
  - 轨迹预处理：验证轨迹并按得分分为成功 / 失败组；
  - 经验提取：从成功 / 失败 / 对比轨迹对中提炼行为洞察；
  - 经验验证：用 LLM 评估经验的质量与可靠性；
  - 向量库更新：将有效经验存入长期存储（用于后续检索）；聚合所有任务的处理结果，得到经验池P_{exp}

$$\mathcal{P}_{\text{exp}} = \bigcup_{g \sim p_{\text{train}}} \Omega_{\text{process}} \left(\left\{ \tau_i^{(g)} \right\}_{i=1}^{N_{\text{pc}}} \right), \quad \text{where} \quad \tau_i^{(g)} \sim \pi_{\theta_{\text{init}}}(\cdot \mid \mathcal{E}, g).$$

- **经验检索（在线阶段）**：针对当前任务g，动态获取相关经验，
  - 将任务query转化为嵌入向量h_q，与经验池中的经验嵌入h_e计算余弦相似度；
  - 取 Top-k 相似经验，经优化模块处理：
    - 重排序：确保经验与任务上下文匹配；
    - 重写：提升经验的通用性与适配性；
  - 最终得到任务对应的经验集合EXP(g)。

$$EXP(g) = \Omega_{\text{refine}} \left(\text{TopK}_k \left(\{ \text{sim}(\mathbf{h}_q, \mathbf{h}_e) \mid e \in \mathcal{P}_{\text{exp}} \} \right) \right),$$

2. Experience-mixed Rollout

虽然以往的经验提供了宝贵的指导，但过度依赖它们可能会阻碍探索。我们引入了一个经验混合阶段，该阶段将非引导和经验引导的轨迹交织在一起。通过整合这两种类型的轨迹，使智能体有效地利用先验知识，同时在新情况下保持灵活性和适应性。

- **轨迹生成**：针对任务g，生成两类轨迹的集合$\mathcal{T} = \mathcal{T}^{(v)} \cup \mathcal{T}^{(e)}$
  - 无引导轨迹$\mathcal{T}^{(v)}$：仅由策略模型自主生成，无外部经验指导；
  - 经验引导轨迹$\mathcal{T}^{(e)}$：将检索到的经验EXP(g)嵌入初始提示词，即 {system_prompt}<EXP>{exp_g}</EXP>{query}
  - 两类轨迹的数量由参数η控制$N_e = \eta \times N$

$$\mathcal{T}^{(v)} = \left\{\tau_i^{(v)}\right\}_{i=1}^{N_v}, \quad \mathcal{T}^{(e)} = \left\{\tau_j^{(e)}\right\}_{j=1}^{N_e}, \quad N = N_v + N_e.$$

- **优势计算**：为统一处理两类轨迹，对所有轨迹计算标准化优势值：

$$\hat{A}(\tau) = \frac{R(\tau) - \mu_R}{\sigma_R + \varepsilon_{\text{norm}}},$$

其中$R(\tau)$是轨迹奖励，μ_R/σ_R是所有轨迹奖励的均值 / 标准差，ε_{norm}保证数值稳定（防止分母为0）

3. Experience Incorporation

为了将检索到的经验的好处无缝地整合到强化学习训练中，我们提出了一种在优化阶段运行的经验整合机制，包括两个互补的组成部分：经验剥离和选择性增强。前者防止了对明确文本线索的虚假依赖，而后者则放大了相关建设性经验的影响。

- **经验剥离：**由于经验引导轨迹的提示词中包含经验文本，若训练时保留这些文本，模型可能“死记硬背”。
  - 做法：在策略梯度计算前，从训练样本中移除经验相关的 token（仅保留系统提示、查询、轨迹）。
  - 作用：让训练仅由轨迹本身驱动，确保模型学习的是内在推理能力，而非依赖外部经验文本。

An illustration of *Experience Stripping*

Before:
`{system_prompt} <EXP>{experience}</EXP> {query}{trajectory}`

After:
`{system_prompt}{query}{trajectory}`

- **选择性增强：**经验剥离后，训练样本（仅依赖查询）与轨迹生成时的样本（依赖查询 + 经验）存在“分布不匹配”，导致重要性比失控，影响训练稳定性。尽管标准PPO/GRPO可以通过限制重要性比来部分缓解这种不稳定性，但过度削弱会抑制经验引导更新的梯度贡献，从而限制学习效率。
  - 做法：对优势值为正的经验引导样本，提高其重要性权重的裁剪阈值，保留这些样本的优化信号。
  - 作用：在保证训练稳定的同时，让模型有效从优质经验轨迹中学习。

06 Self-Attributing

- 现有方法（如 GRPO）依赖 “轨迹级稀疏奖励”，对轨迹中所有动作同等分配功劳，无法区分关键决策与无关步骤，导致样本利用低效、难以从细粒度反馈中学习。
- 为了应对这一挑战，自归因通过利用LLM的推理能力来评估每个动作的贡献，将多步轨迹转化为高效的学习信号。它是代理奖励函数F_{reward}的具体实现，通过从时间信号传播转向基于贡献的归因来重新制定奖励分配，包括过程质量和结果有效性两方面。

- **基于LLM推理的逐步归因**：用于生成过程质量信号，目标是对每个行动对最终结果的贡献进行逐步评估，作为其在任务背景下逻辑正确性的衡量标准。

相比传统人工设计的过程奖励模型：

- 二元标签更易落地，无需复杂的任务专属评分规则；
- 利用 LLM 的上下文推理能力，能灵活做情境化判断；
- 输出的标签是语义合理的 “过程正确性” 衡量，为后续奖励融合提供方向型反馈。

System Prompt for Step-wise Attribution

You are an expert process reward evaluator specializing in attributional analysis.

User Prompt for Step-wise Attribution

TASK DESCRIPTION
<Original task query>

SOLUTION TRAJECTORY (total N steps)

>>> EVAL-STEP 0 <<<
<ACTION> ... <END>
<OBSERVATION> ... <END>

[...continue for all steps...]

OVERALL PERFORMANCE SCORE <value>

REQUIRED OUTPUT FORMAT:

Step 0 Analysis: <your reasoning>
Step 0 Judgment: GOOD/BAD
[...continue for all steps...]

- **基于归因的奖励构建**：把生成的 “GOOD/BAD” 定性标签，转化为步骤级的定量归因奖励r_t^{attr}，作为 “过程导向的密集反馈”，补充传统的“结果导向的稀疏奖励”。
  - **归因标签量化**：GOOD：+1 | BAD：-1
  - **归因信号标准化**：由于轨迹的长度可能会有很大差异，因此简单的步长计算可能会允许更长的轨迹对统计数据产生不成比例的影响，因此为保证数值稳定，先计算每条轨迹的平均归因奖励；再基于所有轨迹的平均奖励，计算整体的均值和标准差

$$\hat{r}_t^{\text{attr}} = \frac{r_t^{\text{attr}} - \mu^{\text{attr}}}{\sigma^{\text{attr}} + \epsilon}, \qquad \hat{r}^{\text{out}} = \frac{R^{\text{out}} - \mu^{\text{out}}}{\sigma^{\text{out}} + \epsilon},$$

- **综合奖励结构**：为了形成完整的学习信号，将基于归因的奖励与基于结果的奖励相结合。这种融合的关键是每个通道的单独归一化，这保留了它们的不同贡献。
  - **结果奖励标准化**：使用Self-Questioning中LLM Judge的结果作为结果奖励，该轨迹级结果奖励在整个训练组中单独归一化。
  - **奖励融合**：每个步骤t的最终复合奖励由 “过程导向的密集归因信号” 与 “结果导向的稀疏信号” 结合而成。结果奖励仅在轨迹的最终步骤应用，以正确归因最终的成功或失败。

$$\hat{r}_t = \alpha \cdot \hat{r}_t^{\text{attr}} + \mathbf{1}_{t=T} \cdot \hat{r}^{\text{out}}.$$

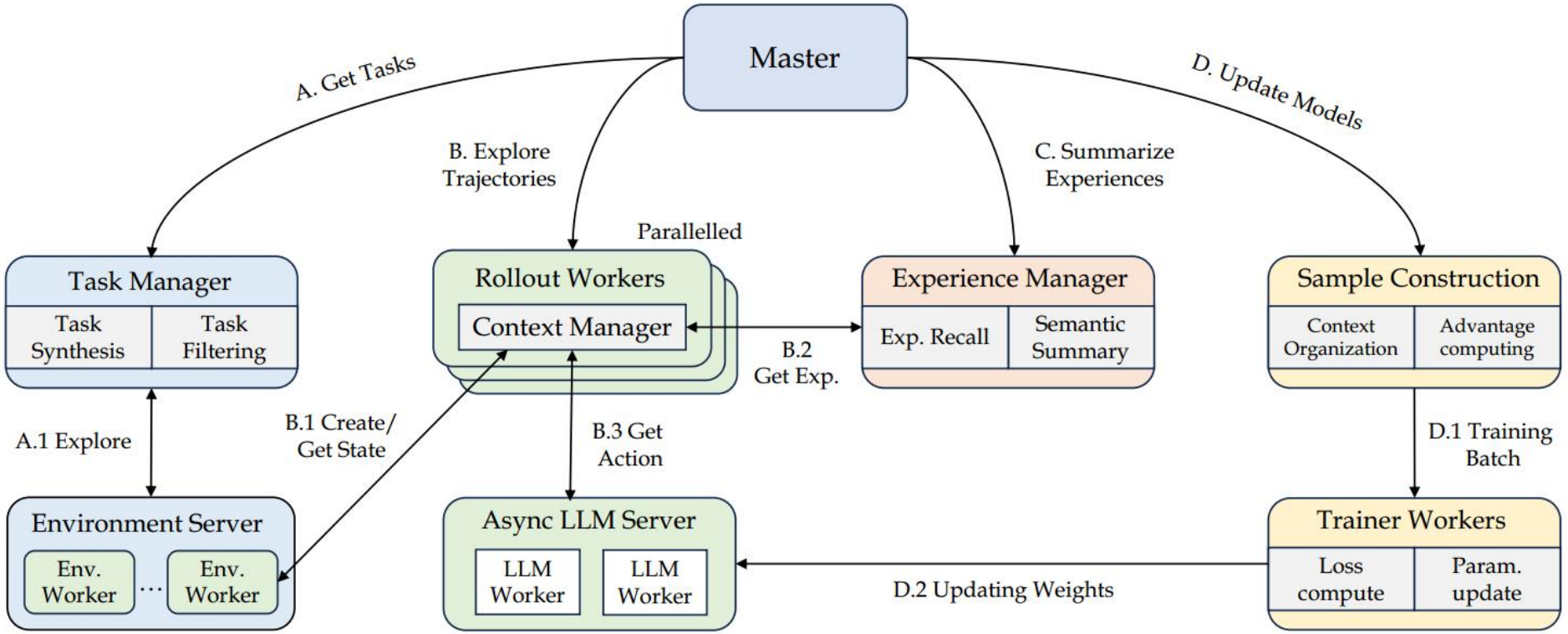
  - **优势计算**：采用一种简化且有效的优势估计方法（参考 DeepSeek-Math 等工作）：将优势直接定义为未折现的累积未来奖励，轨迹中后续所有步骤的权重相等。

$$A_t = \sum_{k=t}^{T} \hat{r}_k.$$

- **策略优化**：将步骤级优势值转化为模型可优化的 token 级信号
  - **步骤到 Token 的映射**：语言模型的策略梯度需要 token 级优势估计，因此将步骤级优势值A_t广播到该步骤生成的所有响应 token 上，得到 token 级优势值A_j^{token}。若 token属于步骤t，则 $A_j^{token} = A_t$
  - **与经验引导训练的整合**：token级优势值会A_j^{token}作为无引导 / 经验引导轨迹的优势估计，融入整体优化目标，该复合优势项同时包含 “归因式过程评估” 与 “结果式任务成功”，让策略同时优化中间步骤质量与最终轨迹结果。

$$\mathcal{L}_{\text{navigating}}(\theta) = -\frac{1}{N}\left[\sum_{i=1}^{N_v} \min\left(r_i^{(v)}\hat{A}_i^{(v)}, \text{clip}\left(r_i^{(v)}, 1-\epsilon_{\text{low}}, 1+\epsilon_{\text{high}}\right)\hat{A}_i^{(v)}\right) + \sum_{j=1}^{N_e} \min\left(r_j^{(e)}\hat{A}_j^{(e)}, \text{clip}\left(r_j^{(e)}, 1-\epsilon_{\text{low}}, 1+\epsilon_j\right)\hat{A}_j^{(e)}\right)\right] + \beta\,\text{KL}\left(\pi_\theta \| \pi_{\theta_{\text{old}}}\right),$$

$$\text{where:}\quad r_i^{(v)} = \frac{\pi_\theta(\tau_i^{(v)})}{\pi_{\theta_{\text{old}}}(\tau_i^{(v)})}, \quad r_j^{(e)} = \frac{\pi_\theta(\tau_j^{(e)})}{\pi_{\theta_{\text{old}}}(\tau_j^{(e)})}, \quad \epsilon_j = \begin{cases} \hat{\epsilon}_{\text{high}} & \text{if } \hat{A}_j^{(e)} > 0 \\ \epsilon_{\text{high}} & \text{otherwise} \end{cases}.$$



AgentEvolver 的整体架构由主协调器驱动**四阶段循环**：

- 任务合成：由任务管理器通过 “自提问” 与环境交互，自动生成候选训练任务；
- 轨迹生成：并行执行任务，多轮交互生成多样轨迹；
- 经验总结：经验管理器将历史轨迹浓缩为经验文本（技能、战术等），用于后续上下文引导；
- 样本构建与模型优化：通过 “自归因” 将轨迹转化为细粒度训练样本，更新策略参数。

08 Context Manager

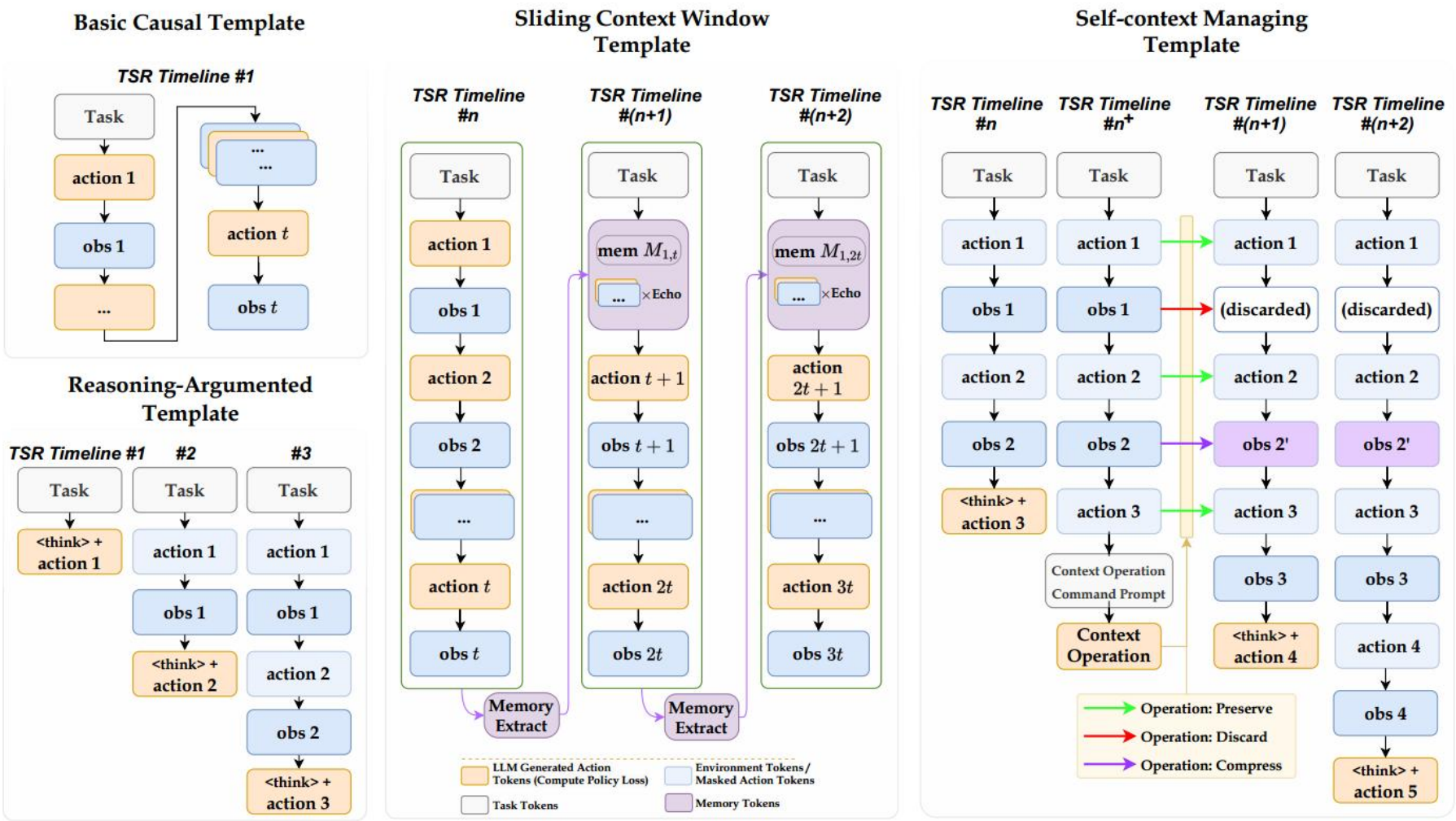
使用LLM进行长期强化学习不仅需要推理和规划，还需要在多个交互中动态管理历史上下文。现有的范式表现出一种权衡：因果多步范式保持了很强的时间一致性，但缺乏灵活性；而与步骤独立多轮范式以巨大的计算开销为代价提供了完全的可编辑性。为了统一这两种观点，AgentEvolver引入了一个**上下文管理器（CM）**，作为在代理与环境交互过程中控制上下文演变的核心基础设施。

其目标是：

1. 保留因果训练的算力效率；
2. 必要时可选择性修改历史上下文；
3. 支持智能体自主管理上下文。

CM 基于两个基础结构实现上下文管理：

- 实时上下文时间线（LCT）：可变序列，作为智能体多轮交互中的 “短期可编辑记忆”；
- 时间线快照记录器（TSR）：不可变缓存，在策略 LLM 生成动作时，保存 LCT 的冻结快照，记录整个交互过程的 token 级信息。



1. **基础因果模板**：严格因果消息组织，时序一致、算力成本低，但上下文不可编辑，内存消耗随长度线性增长；
2. **推理增强模板**：强制 先思考后行动，通过结构化提示让模型生成token，提升复杂决策的推理质量；
3. **滑动上下文窗口模板**：将 LCT 作为滑动窗口，超过阈值时压缩旧内容，保持 GPU 内存稳定，适配超长任务；
4. **自主上下文管理模板**：让智能体主动控制内存，超过 token 限制时，通过提示让 LLM 选择 保留 / 删除 / 压缩消息，实现自主信息过滤。

主要结果：

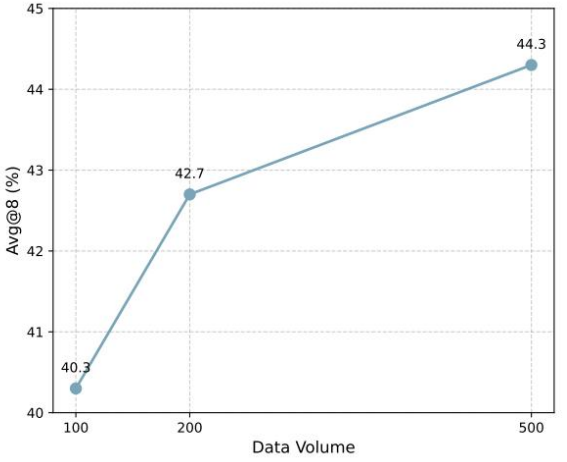
Model	Params	AppWorld		BFCL v3		Avg.	
		avg@8	best@8	avg@8	best@8	avg@8	best@8
Qwen2.5-7B	7B	1.8	5.6	29.8	42.4	15.8	24.0
+Questioning	7B	23.2	40.3	49.0	60.6	36.1	50.5
+Questioning&Navigating	7B	26.3	43.1	53.3	61.0	39.8	52.1
+Questioning&Attributing	7B	25.7	43.7	56.8	65.3	41.3	54.5
AgentEvolver (overall)	**7B**	**32.4**	**51.2**	**57.9**	**69.0**	**45.2**	**60.1**
Qwen2.5-14B	14B	18.0	31.4	41.6	54.1	29.8	42.8
+Questioning	14B	44.3	65.5	60.3	72.1	52.3	68.8
+Questioning&Navigating	14B	45.4	65.3	62.8	74.5	54.1	69.9
+Questioning&Attributing	14B	47.8	65.6	64.9	76.3	56.4	71.0
AgentEvolver (overall)	**14B**	**48.7**	**69.4**	**66.5**	**76.7**	**57.6**	**73.1**

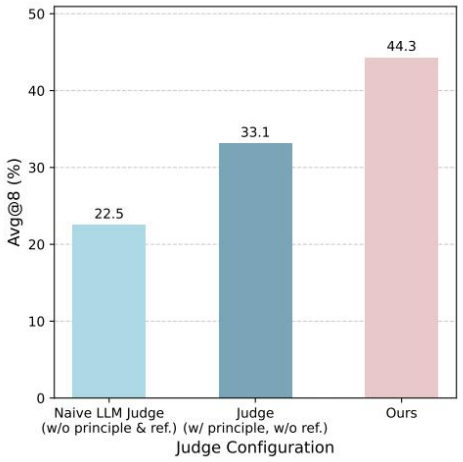
完整的AgentEvolver在所有设置中都达到了最高的性能，这证实了三种机制——自提问、自导航和自归因——相辅相成，以实现框架的全部潜力。

Self-Questioning结果：

- **合成数据的有效性**：合成数据训练的智能体，性能远超零样本，且接近真实数据的训练效果；混合数据训练的智能体，性能完全超过仅用真实数据的情况，说明合成数据能增强多样性、扩展智能体能力边界。
- **合成数据数量的影响**：仅 100 个样本就能让智能体达到 40.3% 的高性能；数据量增加时性能持续提升，但增益逐渐降低，说明合成任务多样性强，少量数据即可实现高效训练，同时降低了对高成本人工数据的依赖。
- **合成数据的能力泛化性**：合成数据训练的智能体能力具有泛化性；性能提升同时包含 “通用能力” 和 “领域专属能力”，为智能体的广泛自演化奠定基础。
- **LLM 裁判的消融实验**：仅用朴素 LLM 裁判时，合成数据难以提升智能体性能；加入 “评分原则 + 参考解” 后，性能显著提升（接近人工数据训练效果），说明 LLM 裁判的设计能提供清晰的评分标准，助力高效训练。

Model	Setting	Appworld		BFCL		Avg.	
		avg@8	best@8	avg@8	best@8	avg@8	best@8
Qwen2.5-7B	Zero-shot	1.8	5.6	29.8	42.4	15.8	24.0
	Original p_{target}	16.1	25.5	58.8	74.0	37.5	49.8
	Synthetic p_{train}	23.2	40.3	49.0	60.6	36.1	50.5
	Hybrid p_{hybrid}	21.8	36.3	65.3	75.6	43.6	56.0
Qwen2.5-14B	Zero-shot	18.0	31.4	41.6	54.1	29.8	42.8
	Original p_{target}	46.1	61.5	68.6	74.3	57.4	68.0
	Synthetic p_{train}	44.3	65.5	60.3	72.1	52.3	68.8
	Hybrid p_{hybrid}	48.4	68.1	73.0	81.1	60.7	74.6



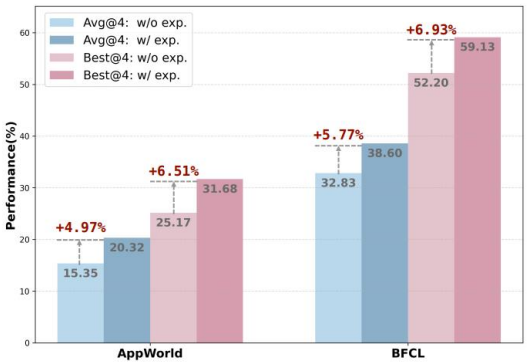


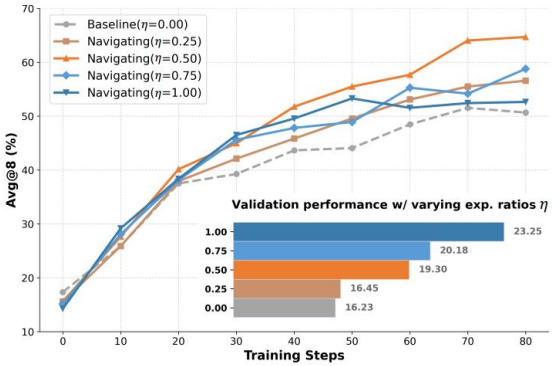
Model	Train on	Appworld		BFCL		Avg.	
		avg@8	best@8	avg@8	best@8	avg@8	best@8
Qwen2.5-7B	Zero-shot	1.8	5.6	29.8	42.4	15.8	24.0
	Appworld	23.2	40.3	**36.1**	**45.0**	29.7	42.7
	BFCL	**1.2**	**4.2**	49.0	60.6	25.1	32.4
Qwen2.5-14B	Zero-shot	18.0	31.4	41.6	54.1	29.8	42.8
	Appworld	44.3	65.5	**56.0**	**68.9**	50.2	67.2
	BFCL	**22.9**	**40.8**	60.3	72.1	41.6	56.5

北京大学 信息工程学院
PEKING UNIVERSITY School of Electronic and Computer Engineering

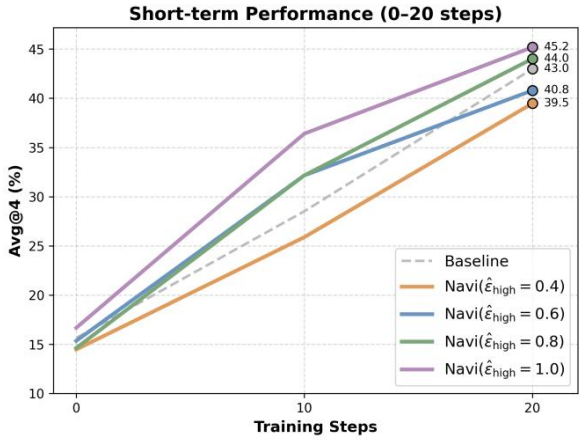
Self-Navigating结果：

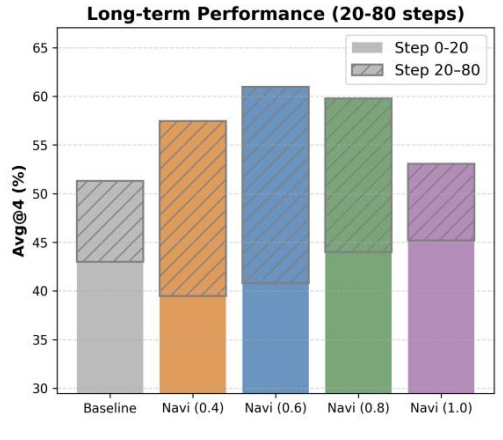
- **经验对轨迹导航的有效性**：推理阶段融入过往经验，可显著提升轨迹质量，无需额外训练即可让模型更高效探索，稳定轨迹生成并提高性能上限。
- **显式 vs 隐式经验学习**：显式学习（推理时通过 ICL 用经验引导）能提升性能，但受限于外部上下文，存在性能天花板；隐式学习（通过 RL 训练让策略内化经验）性能全面超越显式学习和基础 RL；消融实验表明：移除 “选择性增强” 机制会导致性能下降，证明其对有效融合经验信号的必要性。
- **探索与利用的平衡**：高比例经验引导轨迹短期奖励更高，但会限制探索，损害长期优化；中等比例实现探索与利用的最佳平衡，既能借助经验提升短期性能，又能保留足够的探索能力以维持长期优化。
- **短期 vs 长期优化**：过大则加速早期学习，但易过拟合，损害长期性能；设置为0.6：平衡短期快速进步与长期稳定优化，同时保证鲁棒性。





Method	AppWorld		BFCL v3		Avg.	
	avg@4	best@4	avg@4	best@4	avg@4	best@4
w/o RL						
zero-shot	17.3	37.4	33.0	50.6	25.2	44.0
+ exp.	20.2	41.6	41.5	61.9	30.9	51.8
w/ RL						
baseline	51.5	69.8	62.8	73.0	57.2	71.4
Navigating (w/o select)	53.1	63.8	60.3	73.2	56.7	68.5
Navigating (**ours**)	64.7	85.9	65.3	73.9	65.0	79.9

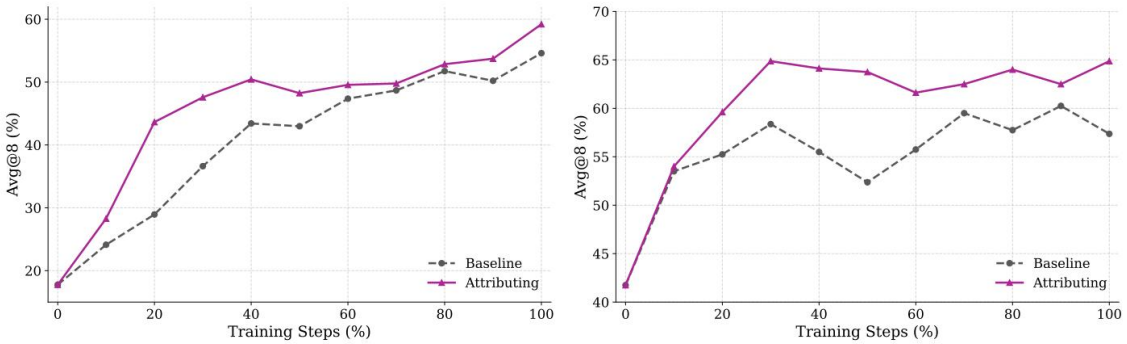




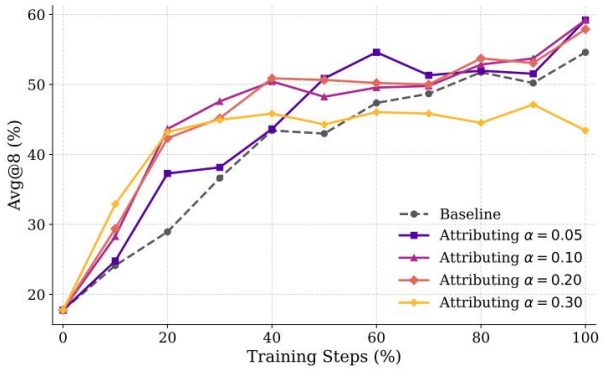
Self-Attributing结果：

- **自归因机制的有效性**：完整的自归因方法在所有设置中性能最优，消融实验证明 “双奖励通道” 的互补性：仅用结果奖励性能合理但功劳分配粗糙；仅用归因奖励虽有提升但不足以支撑任务目标。
- **自归因机制的样本效率**：达到基线 90% 性能所需步骤明显降低；AUC 指标也显著更高。效率提升源于细粒度功劳分配，步骤级归因提供密集反馈，补充结果奖励的稀疏性，让每条轨迹包含更丰富的梯度信息，减少方差与冗余探索。
- **超参数α的分析**：它控制 “归因奖励” 与 “结果奖励” 的权重，存在 “短期收敛” 与 “长期性能” 的权衡：大α（如 0.3）加速早期学习，但易过拟合 LLM 裁判的标签，长期性能下降；小α（如 0.05）早期进步慢，但最终性能与最优相当；最优范围 [0.10, 0.20]平衡密集归因反馈的利用与结果监督的锚定，同时实现快速收敛与稳健的最终性能。

Model	Params	AppWorld		BFCL v3		Avg.	
		avg@8	best@8	avg@8	best@8	avg@8	best@8
Qwen2.5-7B	7B	3.1	9.1	29.8	42.4	16.4	25.7
Attributing	**7B**	**38.4**	**57.1**	**56.8**	**65.3**	**47.6**	**61.2**
- w/o $\hat{r}^{attr}$	7B	33.6	46.6	49.0	60.6	41.3	53.6
- w/o $\hat{r}^{out}$	7B	20.2	37.5	51.3	65.2	35.7	51.4
Qwen2.5-14B	14B	17.8	27.7	41.8	55.3	29.8	41.5
Attributing	**14B**	**59.2**	**75.1**	**64.9**	**76.3**	**62.0**	**75.7**
- w/o $\hat{r}^{attr}$	14B	54.6	71.3	60.3	72.1	57.4	71.7
- w/o $\hat{r}^{out}$	14B	42.5	60.3	63.4	73.5	53.0	66.9



(a) AppWorld (b) BFCL v3



Context Manager 结果：

- **自主上下文管理模板（SCMT）**：整体性能最优，尤其在长 horizon 指标 TGC@8 上表现突出。其优势在于能动态压缩 / 丢弃冗余上下文，让智能体在多工具 API 场景中更聚焦推理，提升交互效率。
- **推理增强模板（RAT）**：性能排名第二，通过先思考后行动的显式步骤，增强了复杂工具环境下的推理能力。
- **滑动窗口模板（SWT）**：性能中等，周期性的记忆总结可能遗漏长序列中的关键依赖，降低了上下文一致性。
- **基础因果模板（BCT）**：作为简单稳健的基线，但缺乏自适应记忆控制，在长上下文任务中完成度较低。

结论：自上下文管理模板最适配复杂长 horizon 任务，而推理增强模板也能有效提升推理能力。

Context-Managing Template (CMT)	TGC		SGC	
	@4 (↑)	@8 (↑)	@4 (↑)	@8 (↑)
Basic Causal Template	0.435	0.506	0.268	0.375
Reasoning-Augmented Template	**0.661**	0.690	**0.500**	**0.571**
Sliding-Window Template	0.560	0.601	0.393	0.411
Self-Context-Managing Template	0.613	**0.720**	**0.500**	**0.607**

请批评指正！

Project: https://arxiv.org/abs/2511.10395
Code: https://github.com/modelscope/AgentEvolver

汪静雅

2025.12.04