

Trace2Skill: 将轨迹局部经验提炼为可迁移的智能体技能

Jingwei Ni^{§,*,2,3}, Yihao Liu^{§,*4}, Xinpeng Liu^{§,*4}, Yutao Sun^{§,*5}, Mengyu Zhou^{†1}, Pengyu Cheng¹, Dexin Wang¹, Xiaoxi Jiang¹ and Guanjun Jiang¹

¹Qwen Large Model Application Team, Alibaba, ²ETH Zürich, ³University of Zurich, ⁴Peking University, ⁵Zhejiang University

*Work done during an internship at Alibaba. †Corresponding author. §Core Contributors.

为大型语言模型 (LLM) 智能体配备领域特定技能, 对于应对复杂任务至关重要。然而, 手动编写技能会带来严重的可扩展性瓶颈。相反, 自动化的技能生成往往产生脆弱或碎片化结果, 因为它要么依赖浅层参数化知识, 要么在非泛化轨迹局部经验上顺序过拟合。为克服这一挑战, 我们提出 Trace2Skill 框架, 其灵感源自人类专家编写技能的方式: 通过整体分析广泛的执行经验, 再将其提炼为单一、全面的指南。与逐个响应个体轨迹不同, Trace2Skill 派遣一组并行的子智能体, 共同分析多样化的执行过程。它提取轨迹特异性经验, 并通过归纳推理, 将这些经验分层整合为统一且无冲突的技能目录。Trace2Skill 既支持深化现有手工编写的技能, 也可从零开始创建新技能。在电子表格、VisionQA 和数学推理等挑战性领域中的实验表明, Trace2Skill 显著优于多种强基准方法, 包括 Anthropic 官方的 `xlsx` 技能。关键在于, 这种基于轨迹的演化并非简单记忆任务实例或模型特有细节: 所演化的技能可在不同规模的 LLM 间迁移, 并泛化至分布外 (OOD) 设置。例如, 由 Qwen3.5-35B 在其自身轨迹上演化出的技能, 使 Qwen3.5-122B 智能体在 WikiTableQuestions 上的表现提升了高达 57.65 个百分点的绝对分数。进一步分析证实, 我们的整体性、并行式整合方式优于在线顺序编辑和基于检索的经验库。最终, 我们的结果表明, 复杂的智能体经验可以被封装为高度可迁移的、声明式技能——无需参数更新, 无需外部检索模块, 仅使用最小规模为 350 亿参数的开源模型即可实现。^a

^a进行中。

1. 引言

基于大语言模型的智能体越来越依赖技能——编码了任务解决流程、领域知识和操作指南的结构化、可复用文档——以在复杂环境中导航 (Anthropic, 2026b)。随着这些智能体被部署到越来越广泛和细致的领域特定用例中, 对高度专业化技能的需求相应增长, 这为手动创建和维护技能带来了可扩展性瓶颈 (Han et al., 2026; Li et al., 2026a; Anthropic, 2026a; Liang et al., 2026)。

即使存在人工编写的技能, 也不能保证它能提升特定智能体、模型或任务分布的性能 (例如, 表1显示, 一个人工专家编写的技能在 SpreadsheetBench-Verified 上使 122B 参数的智能体提升了 +20 个百分点, 反而损害了 35B 参数的智能体) (Ma et al., 2024)。

这些压力促使针对特定用例自动创建和适配技能 (Han et al., 2026)。

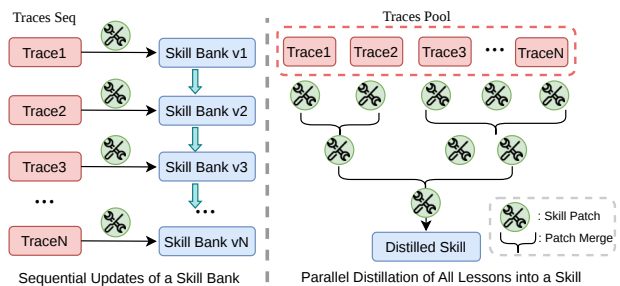


Figure 1 | 左图: 并行工作的在线情景, 其中新到达的迹序列地演化出技能库。右图: Trace2Skill 并行分析一组迹, 并分层整合经验以推导出可泛化的标准操作流程。

然而, 仅依赖大语言模型的参数化知识来合成技能所带来的收益有限, 即使使用最先进的专有模型也是如此, 主要原因在于参数化知识缺乏目标领域具体细节和常见陷阱的信息 (Li et al., 2026b; Jiang et al., 2026)。为解决这一问题, 相关工作提出在在线情景下利用智能体执行经验来提升技能, 其中智能体持续与环境交互, 并根据接收到的轨迹不断演化其技能集合 (Yang et al., 2026; Xia et al., 2026a; Alzubi et al., 2026; Zhou et al., 2026; Jiang et al., 2026)。

尽管这种连续、在线的范式展现出潜力, 我们从一个不同的角度来解决技能演化问题——这一角度更贴近人类专家创作技能的方式。具体而言, 我们观察到现有的在线范式在两个关键方面往往与人类的方法相偏离:

- **技能碎片化与整合:** 现有工作通常创建新的、针对特定轨迹的技能, 导致技能集合庞大, 可能引发检索困难 (Li, 2026)。相比之下, 人类专家通常为每个领域构建单一的综合性技能, 包含广泛的流程指导和错误预防检查清单。
- **顺序更新与整体更新:** 在在线设置中, 技能通过孤立的流入轨迹中的课程进行顺序更新 (Jiang et al., 2026; Xia et al., 2026a)。这模拟了作者在逐步学习某一领域时持续编辑技能的情景, 但在获得充分的领域特定知识之前就过早作出反应。相反, 人类专家会在将领域知识具体化为技能之前, 先建立全面且高层次的领域理解。图 1 说明了这些对比。

受这些观察的启发, 我们提出了 Trace2Skill, 一个旨在模拟人类整体化方法的框架。与逐个响应轨迹不同, Trace2Skill 并行分析大量轨迹局部经验, 并将其中的共性模式提炼为单一、全面的智能体技能。Trace2Skill 分为三个阶段: (1) **轨迹生成:** 一个智能体在不断演化的任务集合上并行运行, 生成一批执行轨迹。(2) **并行多智能体补丁提议:** 一组成功与错误分析子智能体独立处理轨迹批量, 针对技能提出针对性补丁。(3) **无冲突整合:** 子智能体提出的补丁按层级合并为技能目录的连贯更新, 在每一步均利用程序化冲突检测和格式验证确保一致性。

我们在整合过程中同时处理所有补丁, 主要有两个原因。首先, 这一过程起到了归纳推理的作用 (Xiong et al., 2025; Li et al., 2025; Lin et al., 2025), 从经验特定的补丁中挖掘可泛化的模式, 建立起对领域的高层次理解, 类似于人类专家的先验知识。其次, 并行分析大量轨迹带来了显著的效率优势, 并确保了对领域的全面视角。这体现了智能体集群 (Kimi Team, 2026) 的核心设计智慧, 即通过并行化的子智能体高效处理多个信息源。该框架支持两种模式: 深化现有手工编写的技能, 以及从无效的大型语言模型生成草稿出发, 从零开始创建有效的技能。

这项工作最令人惊讶的发现不仅是轨迹分析能够提升技能质量, 而且它在不牺牲泛化能力的前提下实现了这一点。尽管对特定任务分布和特定大语言模型的轨迹进行了深入分析, 但进化后的技能在不同模型规模间仍可迁移 (例如, Qwen3.5-35B (Team, 2026) 进化出的技能可提升 Qwen3.5-122B), 并能泛化到分布外任务领域 (例如, 从电子表格编辑到维基百科表格问答)。分析将这种可迁移性归因于从广泛轨迹中成功挖掘出的普遍存在且高度有用的模式。这一发现挑战了普遍假设, 即经验本质上是模型和任务特定的, 并且必须通过情景记忆的检索进行管理 (Ouyang et al., 2026; Wang et al., 2024; Qian et al., 2024; Nottingham et al., 2024; Liu et al., 2025)。相反, 我们表明经验可以提炼为可迁移的陈述性技能。我们进一步验证了 Trace2Skill 在为数学和视觉推理创建有用技能方面的有效性。

进一步分析表明, Trace2Skill 在经验学习的其他流行范式中表现更优: (1) Reasoning Bank (Ouyang et al., 2026), 该方法首先从每条轨迹中保存可泛化的经验教训, 并在推理时根据任务相似度检索有用的经验。(2) 一种在线设置, 新轨迹依次到来, 技能根据新学到的经验不断演化。关键在于, Trace2Skill 创建或加深的技能完全不需要外部检索模块, 因此可以在更广泛的智能体-技能生态系统中无缝移植。

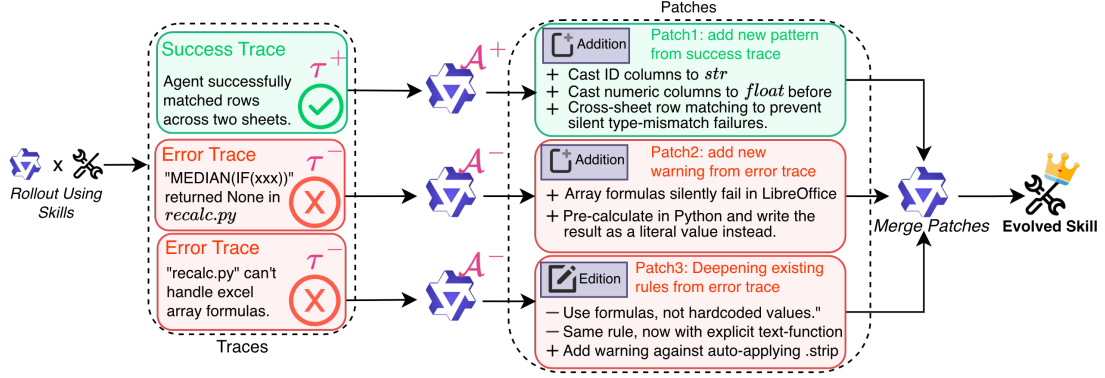


Figure 2 | Trace2Skill 的三阶段流水线概述。**阶段 1**：一个冻结的智能体 π_θ 在不断演化的集合上使用初始技能 S_0 （人工编写或由大语言模型起草）进行滚动执行，生成标注轨迹 \mathcal{T}^- （失败）和 \mathcal{T}^+ （成功）。**阶段 2**：并行的错误分析与成功分析器独立处理各个轨迹，并提出技能修补方案。**阶段 3**：所有修补方案通过归纳推理并结合程序化冲突预防机制合并为单一整合更新，生成性能与泛化能力均得到提升的演化技能 S^* 。

- Trace2Skill，一种支持对现有手工编写的技能进行深化以及从零开始创建新技能的自动技能生成与适应框架。通过采用完全并行化的补丁提议和无冲突的整合方式，Trace2Skill 模拟了人类技能编写过程：在起草综合性技能之前，通过广泛的轨迹分析建立广泛的先验知识 (§2)。
- 实证证据表明，基于轨迹的演化能够生成高质量、可泛化的技能，这些技能在不同规模的大型语言模型之间以及分布外的任务领域中均能有效迁移 (§3)。
- 一个演示表明，开源的小规模大模型（例如 350 亿参数）足以实现稳健的技能演化，从而消除了在同期工作中所依赖的专有模型 (§3)。
- 进一步分析表明，并行化优于顺序的在线技能更新；单一综合技能优于基于检索的推理库；Agentic 错误分析优于普通的基于大语言模型的分析 (§4)。

2. Trace2Skill

图 2 展示了 Trace2Skill 的三阶段流水线。我们首先形式化技能结构和演化目标 (§2.1)。第 1 阶段、第 2 阶段和第 3 阶段分别在 §2.2、§2.3 和 §2.4 中详细描述。

2.1. 技能与问题形式化

一个技能 S 是一个结构化的、人类可读的知识目录，由一个根 Markdown 文档 M (SKILL.md) 和一组辅助资源 $\mathcal{R} = \{r_1, \dots, r_K\}$ 组成：

$$S = (M, \mathcal{R}), \quad \mathcal{R} = \{\text{scripts, references, assets}\}. \quad (1)$$

M 将过程性知识以自然语言形式编码：何时应用某种技术、分步策略以及已知的失败模式。辅助资源提供用于确定性任务的可执行脚本以及上下文或领域特定的参考信息。

技能演化问题的形式化。 令 π_θ 表示一个具有固定参数 θ 的基于大语言模型的智能体，在推理时配备了一个前置技能 S 。令 $\mathcal{D}_{\text{evolve}}$ 和 $\mathcal{D}_{\text{test}}$ 为从可能不同的分布中抽取的不相交的任务集合。我们定义成功率如下

$$\mathcal{P}(S; \pi_\theta, \mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{t \in \mathcal{D}} 1[\pi_\theta(t; S) = y_t^*], \quad (2)$$

其中 y_t^* 是任务 t 的真实答案。技能演化的目标是基于 $\mathcal{D}_{\text{evolve}}$ 上的轨迹构建一个改进的技能，而无需更新 θ ，使得：

$$\mathcal{S}^* = \mathcal{E}(\mathcal{S}_0, \mathcal{D}_{\text{evolve}}; \pi_\theta), \quad \mathcal{P}(\mathcal{S}^*; \pi_\theta, \mathcal{D}_{\text{test}}) > \mathcal{P}(\mathcal{S}_0; \pi_\theta, \mathcal{D}_{\text{test}}). \quad (3)$$

我们研究了 \mathcal{S}_0 的两种初始化方式：一种是由人类专家编写的技能（深化模式），另一种是仅基于参数化知识由大语言模型生成的草稿（创造模式），这反映了 Trace2Skill 在现实世界中的两种主要应用场景。

2.2. 第一阶段：轨迹生成

我们采用 ReAct (Yao et al., 2023) 作为智能体框架。给定 \mathcal{S}_0 ，我们在每个任务 $t_i \in \mathcal{D}_{\text{evolve}}$ 上运行 π_θ ，查询为 q_i ，生成一条轨迹：

$$\tau_i = \pi_\theta(q_i; \mathcal{S}_0) = (q_i, (r_1^{(i)}, a_1^{(i)}, o_1^{(i)}), \dots, (r_{T_i}^{(i)}, a_{T_i}^{(i)}, o_{T_i}^{(i)}), y_i), \quad (4)$$

其中 $r_k^{(i)}$ 表示第 k 条推理迹线， $a_k^{(i)}$ 表示工具调用， $o_k^{(i)}$ 表示观察结果， $y_i \in \{0, 1\}$ 表示正确性结果。语料库 $\mathcal{T} = \{\tau_1, \dots, \tau_N\}$ 被划分为：

$$\mathcal{T}^- = \{\tau_i \in \mathcal{T} : y_i = 0\}, \quad \mathcal{T}^+ = \{\tau_i \in \mathcal{T} : y_i = 1\}. \quad (5)$$

轨迹生成是完全可并行化的；在实际应用中，使用一个 1220 亿参数的大型语言模型生成 200 条包含 50 多个弯道的轨迹，所需时间不足 2 个 GPU 小时。智能体系统提示模板见附录 B.1。

2.3. 第二阶段：并行多智能体补丁提议

一组专门的分析智能体子代理，每个被分配到一条单一轨迹 τ_i ，独立提出对技能的修改建议。每个分析智能体获取 \mathcal{S}_0 的冻结副本和一条轨迹，并输出一个技能补丁：

$$p_i = \begin{cases} \mathcal{A}^-(\mathcal{S}_0, \tau_i), & \tau_i \in \mathcal{T}^- \\ \mathcal{A}^+(\mathcal{S}_0, \tau_i), & \tau_i \in \mathcal{T}^+ \end{cases} \quad (6)$$

所有分析员被同时分派至线程池，生成补丁池 $\mathcal{P} = \{p_i\}$ ，各智能体之间无顺序依赖关系。两个角色均被指示提出能够泛化至单一观测轨迹之外的补丁，并严格遵循 Anthropic 关于简洁性、可操作性和层级披露的技能写作风格建议 (Anthropic, 2026a)。由于我们假设不存在更强的教师模型，错误比成功更难诊断，这促使采用非对称的分析员设计。

成功分析员 (\mathcal{A}^+)。 \mathcal{A}^+ 遵循固定的单次执行 workflow：它清理轨迹，识别对正确答案有贡献的可泛化行为模式，并提出技能修补方案。单次调用的设计既足够又高效，因为成功的轨迹无需交互式诊断。

错误分析员 (\mathcal{A}^-)。 \mathcal{A}^- 以 ReAct 风格的多轮智能体环形式实现。给定 $\tau_i \in \mathcal{T}^-$ ，它可以检查完整迹，读取输入/输出文件，并将智能体的答案与真实值进行对比——通过迭代方式逐步缩小根本原因，进而提出修复方案。当 \mathcal{A}^- (1) 成功修复并因果性地解释失败，或 (2) 耗尽其轮次预算时，该环终止。若上述任一条件均未产生有效的因果分析，则 τ_i 会被排除在补丁池之外。这一质量门控机制确保 \mathcal{P}^- 中的每个补丁均基于已验证的故障原因，与以往通过单次非交互式大模型调用推导见解的方法 (Ouyang et al., 2026) 形成对比。第 §4.3 节展示了智能体与仅使用大模型的错误分析方法之间的消融对比。

补丁提议的独立性。所有分析员均在 \mathcal{S}_0 的冻结副本上操作，无法观察到其他智能体的补丁。这种独立性可防止过早收敛，从而在 \mathcal{P} 中保持每条轨迹观测结果的完整性多样性。分析员提示模板和代表性示例补丁见附录 B.2。

2.4. 第三阶段：无冲突补丁合并

令 $\mathcal{P} = \mathcal{P}^- \cup \mathcal{P}^+$ 为第 2 阶段的完整补丁池。第 3 阶段将 \mathcal{P} 合并为单一连贯的技能更新 p^* ，并将其应用于 \mathcal{S}_0 ，共同实现两个目的：冲突消除与归纳泛化。

分层合并与程序化冲突预防。这些补丁按照 $L = \lceil \log_{B_{\text{merge}}} |\mathcal{P}| \rceil$ 层级的层次结构进行合并 ($L \leq L_{\text{max}}$)。在每一层级 ℓ ，最多 B_{merge} 个补丁被合成一个统一的补丁：

$$p^{(\ell+1)} = \mathcal{M}(\pi_\theta, \mathcal{S}_0, \{p_1^{(\ell)}, \dots, p_{B_{\text{merge}}}^{(\ell)}\}), \quad (7)$$

其中 $\mathcal{M}(\pi_\theta, \mathcal{S}_0, \cdot)$ 用于去重、解决冲突并保留唯一见解。至关重要的是， \mathcal{M} 重用了生成轨迹和提议补丁时所用的同一 π_θ ——使整个流水线具有自包含性：单一 LLM 收集经验，分析经验，并将其提炼为改进后的技能，无需外部教师。最终的 p^* 被转换为差异风格的编辑操作，并以程序化方式应用。三个确定性防护机制确保正确性：(1) 引用不存在文件的补丁将被拒绝；(2) 针对同一文件中相同行范围的编辑会被标记为冲突并暂存；(3) 更新后的 \mathcal{S} 将通过技能格式检查器进行有效验证。

补丁整合作为一种归纳推理。除了冲突消除之外， \mathcal{M} 的分层应用在补丁池上执行了归纳推理。由于每个 p_i 均源自单一轨迹， \mathcal{P} 整体编码了 π_θ 在不断演化的集合中所表现出的行为分布。 \mathcal{M} 被明确指示识别普遍模式——即在独立补丁中持续出现的编辑——因为跨越多种轨迹的重复观察更有可能反映系统的任务特性，并能泛化到未见过的任务以及不同的智能体模型。相反，仅在少数或单一补丁中出现的编辑被视为可能具有特定性而被舍弃。这种基于普遍性的加权整合机制，正是深度逐轨迹分析生成可泛化技能的方式。

进化后的技能 $\mathcal{S}^* = (\mathcal{M}^*, \mathcal{R}^*)$ 替换了 \mathcal{S}_0 ，且在推理时可直接使用，无需任何检索索引。合并操作符提示模板和一个合并补丁示例 p^* 见附录 B.3。

2.5. 两种演化模式

技能深化。 \mathcal{S}_0 由人工专家编写的技能初始化。流水线通过添加来自 \mathcal{T}^- 的失败特定指导并强化 \mathcal{T}^+ 的有效策略来优化 \mathcal{S}_0 。

从零开始创造技能。 \mathcal{S}_0 由 π_θ 从参数化知识中单独绘制的技能初始化，不访问任务轨迹。正如我们在 §3 中所示，此草稿对无技能状态 — $\mathcal{P}(\mathcal{S}_0; \pi_\theta, \mathcal{D}_{\text{test}}) \approx \mathcal{P}(\emptyset; \pi_\theta, \mathcal{D}_{\text{test}})$ — 没有显著改进，因此从这一点开始的演化构成了真正的技能创造：该流水线从一个性能中立的初始化中产生了一个有用的技能，完全由轨迹证据驱动。

3. 实验

3.1. 实验设置

数据集与技能。我们的主要实验聚焦于电子表格领域，该领域要求智能体与文件系统交互并操作 xlsx 文件，其内容在缺乏结构化工具的情况下难以检查。我们使用 SpreadsheetBench-Verified (Ma et al., 2024)，将其 400 个样本分为 200 个用于演化集，200 个保留用于测试；演化过程中不会观察到任何测试样本。此外，我们在完整的 SpreadsheetBench 上报告了软（子问题通过率）和硬（所有子问题必须通过）得分。对于分布外 (OOD) 泛化能力评估，我们在 WikiTableQuestions (Pasupat & Liang, 2015) (WikiTQ) 上进行测试，其数据来源（维基百科）和任务类型（组合语义解析）均有所不同；输入和期望输出被转换为电子表格格式，使得 xlsx 技能无需修改即可直接应用。所有结果均基于三个随机种子 (41、42、43) 的平均值，并采用各基准的官方评估标准。

比较了两种基准技能：(1) Anthropic 官方的 `xlsx` 技能 (**人工编写**)，由高质量的人工专家编写；以及 (2) 仅通过参数化知识由 Qwen3.5-122B-A10B 提示生成的 `xlsx-basic` 技能 (**参数化**)，仅包含常识级别的任务描述，无轨迹锚定（详见附录 B.1）。

技能设置。我们评估六种条件：**无技能**（无技能文档），**人工撰写** (`xlsx`)，**参数化** (`xlsx-basic`)，**+ 错误**（仅包含错误分析者的 Trace2Skill），**+ 成功**（仅包含成功分析者的 Trace2Skill），以及 **+ 综合**（同时包含两类分析者的 Trace2Skill）。技能深化从人工撰写开始；技能创建从参数化开始。

实现细节 Trace2Skill 采用端到端自演化机制：同一 LLM 同时担任轨迹生成器、补丁提议者和技能编辑器的角色。我们实验了两个 Qwen3.5 MoE 模型：Qwen3.5-122B-A10B 和 Qwen3.5-35B-A3B。两者均为指令/思考混合模型；我们使用指令模式完成多轮 ReAct 风格的 Agentic 任务，使用思考模式完成单次调用任务（层次化合并、成功分析、补丁转换）。模型通过 vLLM (Kwon et al., 2023) 提供服务，并采用推荐的 Qwen3.5 生成配置¹。第一阶段每道题生成 1 条轨迹。在第二阶段，128 个子智能体并行运行，我们使用 32 的合并批量大小。对于所有 ReAct 风格的智能体，我们将交互轮次预算设置为 100。

3.2. 主要结果

表 1 展示了所有技能条件、模型规模和迁移方向下的实验结果。我们报告进化后技能相对于其对应基准的表现差异：将技能深化与现有手工编写的技能进行对比，将技能创建与模型的基础参数性能进行对比。我们采用 **Avg** 作为主要汇总指标：一个真正对智能体有益的技能应当在不同模型规模和任务领域之间均能有效迁移，因此 Avg 平等权重地考虑分布内 `SpreadsheetBench` 性能 (Vrf/Soft/Hard, 两种模型规模) 以及 WikiTQ 迁移性能 (两种模型规模)，奖励泛化能力而非分布内特化。

人工编写是一种强大的手工先验，但不具备可移植性；参数化方法则较弱。人类编写的基准对 122B 智能体表现强劲，在 `SprBench-Vrf` 上达到 48.33%，在 WikiTQ 上达到 74.68%，但其在不同模型规模间迁移效果不佳：对于 35B 智能体，其在 `SprBench-Vrf` 上比 No Skill 低 - 9.3 pp，在 WikiTQ 上低 - 4.3 pp。相比之下，参数化基准整体上始终接近 No Skill (122B 智能体在 `SprBench-Vrf` 上为 26.17% vs. 27.67%)，证实仅靠参数化知识无法产生有用的技能内容 (Han et al., 2026)。这两个基准促使我们提出深化 (Deepening) 与创造 (Creation)：前者探讨是否可以优化一个强大的人工先验，后者则探讨是否可以通过轨迹基础的蒸馏从一个不足的技能出发构建出有用的技能。

深化可靠地增强了人类编写技能在分布内电子表格任务上的表现。从人工编写开始，由 122B 模型撰写的深度化策略在 `SprBench-Vrf` 上分别取得 +Error 时 + 17.5 pp 和 +Combined 时 + 21.5 pp 的提升，同时提升了 Soft 与 Hard 得分。这些提升并不仅限于撰写模型本身：由 35B 模型撰写的深度化策略在 +Error 情况下，当由 122B 智能体使用时，可在 `SprBench-Vrf` 上获得 + 16.7 pp 的提升；而由 122B 模型撰写的对应策略则使 35B 智能体在相同任务上获得 + 27.0 pp 的提升。同样的优化技能也超越了训练分布，在 WikiTQ 上，由 122B 模型撰写的深度化策略分别带来 + 1.6 pp (+Error) 和 + 4.6 pp (+Combined) 的改进。

生成效果显著优于弱参数基准，能够达到甚至超过人类写作的质量。由于参数化方法是一个较差的起点，相关比较在于蒸馏技能是否能从零开始恢复有意义的能力。答案是肯定的：由 1220 亿参数作者创作的 Creation +Error 在 `SprBench-Vrf` 上获得了 + 22.8 pp 的提升，使性能接近人类撰写水平，尽管起点较弱。在某些情景中，Creation 甚至优于人类撰写：350 亿参数作者创作的 Creation +Error 技能，当由 1220 亿智能体使用时，在 WikiTQ 上相比参数化方法取得了 + 57.7 pp 的提升（表中最佳，达到 81.38%），并超出人类撰写水平 + 6.7 pp。

¹ 参见 <https://huggingface.co/Qwen/Qwen3.5-35B-A3B> 和 <https://huggingface.co/Qwen/Qwen3.5-122B-A10B>。

Avg 列标识了在分布内、分布外及跨模型使用情况下均表现稳健的设置。从平均得分来看，表现最强的配置是那些能够同时提升表格中多个片段的方案，而非仅在单一基准或技能用户模型上出现峰值。

最佳平均得分来自“创作”情景，其中由 350 亿参数模型撰写的“创作 + 错误”达到 + 18.3 pp，而由 1220 亿参数模型撰写的“创作 + 成功”达到 + 17.6 pp，表明从零开始的技能合成在跨数据集和跨用户模型平均后仍能保持强劲表现。

与此同时，“深化”情景整体保持竞争力，而“综合”情景尤为突出，因其在所有四种作者-模式组合中均保持较高得分，而非依赖于某一特定有利情景。

在各类分析师中，**+Combined** 始终是最具持续性的强烈信号，**+Error** 是最可靠的信号，而 **+Success** 则波动最大。以相对于相应基准的平均提升衡量，**+Combined** 是最稳定的表现优异者，而 **+Error** 在每种情景下均保持可靠正值，可作为最安全的默认信号。相比之下，**+Success** 的方差最高：它在单一情景下的平均增益最大（122B 作者创作任务中为 + 17.6 pp），但也是唯一一个在某些情景下低于基准的情况（122B 作者深化任务中为 - 0.9 pp）。这一模式表明，基于成功的补丁可能具有极高价值，但仅当层级合并机制能有效过滤它们时；否则，其稳定性不如基于错误的更新，这促使我们设计更为选择性的成功分析器（参见 §6）。

3.3. 数学推理

为了评估 Trace2Skill 是否能够泛化到电子表格之外，我们使用 **DAPO-Math-Train-400** 作为演化集合，将其应用于数学推理。

我们在 **DAPO-Math-Test-100**（分布内；通过率%）和 **AIME 2026**（分布外竞赛数学；30 道题平均得分 @8）上进行评估。遵循第 3.2 节的跨模型协议，我们使用错误分析器从零开始构建技能。结果如表 2 所示。

蒸馏出的数学技能在不同模型和基准上带来持续的提升（表 2）。122B 生成的技能提升在 DAPO-Math-Test-100 上提升了 + 3.0 个百分点，在 AIME 2026 上提升了 + 2.9 个百分点（同模型），并且跨模型正向迁移：对于 35B 智能体，在 DAPO-Math-Test-100 上提升了 + 5.0 个百分点，在 AIME 2026 上提升了 + 5.0 个百分点。35B 生成的技能同样有效：在 DAPO-Math-Test-100 上同模型提升了 + 4.0 个百分点，跨模型提升了 + 2.0 个百分点，证实了基于轨迹的蒸馏具有领域无关性，并且能扩展到竞赛级别的评估。

Table 2 | Math reasoning results shown as deltas from the No Skill baseline. **D-Test**: DAPO-Math-Test-100 pass rate (%); **AIME**: AIME 2026 avg@8 over 30 problems (%). Reference row remains absolute; evolved rows use green / red delta intensity.

Condition	Skill User: 122B		Skill User: 35B	
	D-Test↑	AIME↑	D-Test↑	AIME↑
No Skill	92.0	90.4	89.0	83.3
122B-Authored +Error	+3.0	+2.9	+5.0	+5.0
35B-Authored +Error	+2.0	+1.3	+4.0	+0.5

3.4. 视觉问答系统

为了评估 Trace2Skill 在多模态视觉推理任务上的泛化能力，我们将其应用于**视觉问答(VQA)**，以 **DocVQA Mathew et al. (2020)** 作为目标基准。

DocVQA 要求联合理解文档图像——表单、表格、发票、信函、报告——并通过对视觉和文本元素的提取、定位与推理来回答自然语言问题。

我们使用官方验证集（5,349 个问题-图像对），将前 2,700 个实例作为演化集，其余 2,649 个作为保留的评估集。

我们报告 **ANLS**（平均归一化莱文斯坦相似度，官方指标）和**准确率**（ $\text{ANLS} \geq 0.5$ ，%）。

与第 3.2 节类似，我们通过错误分析器从头构建技能。结果如表 3 所示。

表 3 展现了一个细致入微的图景。无技能行揭示了一个出人意料的反转：在没有技能的情况下，35B 智能体 (0.6843 ANLS) 的表现优于 122B 智能体 (0.6424 ANLS)，在 DocVQA 上表现更佳。

尽管在任务性能上具有优势，35B 模型作为技能的作者却表现不佳。由 35B 模型撰写的技能对 122B 模型带来的提升微乎其微 (+0.009 ANLS)，且反而会显著降低同模型 35B 的表现 (-0.062 ANLS, -6.2 pp 准确率)。相比之下，由 122B 模型撰写的技能在同模型上带来了 +0.1639 ANLS 和 +15.3 pp 准确率的提升，并且同样能有效迁移到 35B 模型上 (+0.1554 ANLS, +13.6 pp 准确率)。这种差异表明，技能创作中的归纳推理——即在轨迹中识别重复出现的失败模式，并将其提炼为可执行规则——是一种独立于任务执行的能力。一个在 DocVQA 任务中表现良好的模型，并不一定具备反思能力来分析为何会失败，也无法将这些观察泛化为可迁移的技能。

Table 3 | DocVQA results shown as deltas from the No Skill baseline (evaluation set: 2,649 instances). **ANLS**: Average Normalized Levenshtein Similarity; **Acc**: $\text{ANLS} \geq 0.5$ (%). Reference row remains absolute; evolved rows use green / red delta intensity.

Condition	Skill User: 122B		Skill User: 35B	
	ANLS↑	Acc↑	ANLS↑	Acc↑
No Skill	0.6424	71.2	0.6843	75.2
122B-Authored +Error	+0.1639	+15.3	+0.1554	+13.6
35B-Authored +Error	+0.0093	+0.9	-0.0620	-6.2

4. 分析

4.1. 并行整合与串行编辑

在在线技能演化范式中，随着新轨迹批量的到达，技能会依次进行更新。我们通过与两种顺序基准方法对比，分离出并行、多对一整合的贡献：**Seq-B=1**，即在每一条轨迹后更新技能；以及 **Seq-B=4**，即在每四个轨迹的批量后更新技能。三种条件均仅使用误差分析器，并从人工编写的技能初始化；结果如表 4 所示。

在 122B 模型上，平行整合在所有 SpreadsheetBench 指标上均优于两种顺序设置 (+4.0 pp Vrf 高于 Seq-B=1, +6.8 pp 高于 Seq-B=4)。在 35B 模型上，平行设置在 Vrf 上表现更优，但顺序设置-B=1 在 Soft 和 Hard 指标上得分略高，表明较小的模型在某些条件下可能从更多增量更新中受益。然而，这种微小的质量差异带来了 20× 的计算成本，且随着分析轨迹数的增加，效率差距呈线性扩大。

延迟。使用 $W=128$ 个工作者和 $N \approx 70$ 个错误课程，所有分析员在单个并行轮次中执行；层级合并增加了 $\lceil \log_2 N \rceil \approx 7$ 个额外的串行轮次（每层合并一个），总共产生 ≈ 8 个串行 LLM 调用轮次。串行基准需要分别 N 和 $\lceil N/B \rceil$ 个轮次，因为每个技能编辑都依赖于前一个。实际上，这相当于 3 分钟（并行）对比 60 分钟（Seq-B=1, 20×）和 15 分钟（Seq-B=4, 5×），且时间差距随 N 线性增长。所有时间均以 8-GPU A800 结点的结点小时为单位。

除了效率之外，并行合并还具有结构上的优势：所有补丁提议均源自同一个冻结的初始技能 S_0 ，从而避免了在线设置中固有的序列漂移问题，因为在在线设置中，每次技能更新都会改变后续轨迹分析的上下文。层次化合并随后对全部轨迹局部观测结果进行同步的归纳推理，选择在多种轨迹中反复出现的模式，而非仅在最近更新中反复出现的模式。

4.2. 迹 2 技能与检索-记忆基准

ReasoningBank (Ouyang et al., 2026) 将每个轨迹中提炼出的可泛化经验进行存储，并在推理时利用任务-查询相似度检索最相关的记忆。由于它同时借鉴了成功与失败的轨迹，我们将其与 **+Combined** 进行对比，后者使用相同的轨迹池，但将轨迹信息提炼为可移植的技能文档，而非维护一个检索索引。我们采用官方提示和推荐的检索设置（取 top 1）实现了基于 ReasoningBank 风格的检索基准，以 Qwen3-Embedding-8B 作为检索器。相同模型下的深化结果如表 5 所示。

+Combined outperforms ReasoningBank by large margins: +13.8 pp Vrf, +7.1 pp Soft, +8.2 pp Hard for the 122B model; +9.2 pp Vrf, +1.5 pp Soft, +0.8 pp Hard for 35B. 在 35B 模型上的 ReasoningBank (20.50% Vrf) 仅略高于无技能基准 (19.00%)，表明当 35B 模型的查询表示与存储的记忆嵌入不匹配时，检索器无法有效提取相关指导。

我们把性能差距归因于三个因子。首先，检索质量对测试查询与存储记忆键之间的表面相似度非常敏感：当测试分布的表述或结构与不断演变的集合不同时，检索效果会下降，而蒸馏后的技能则无需修改即可迁移。其次，检索到的片段会与任务上下文争夺模型的注意力，而预先加载到系统提示中的技能在看到任何特定任务内容之前就已经被整合。第三，Trace2Skill 的层级合并机制能够主动去重并从轨迹局部观察中抽象出通用原则；而检索库中的原始轨迹摘要并未经过冗余性或泛化能力的筛选。综上所述，这些结果支持一个前提：将轨迹证据蒸馏为紧凑且模型无关的技能文档，比采用情景式检索更能有效利用轨迹证据。

4.3. Agentic 错误分析与单次 LLM 调用基准对比

许多先验工作通过单次非交互式大语言模型调用从错误轨迹中提取可迁移的教训或技能 (Ouyang et al., 2026; Xia et al., 2026a; Yang et al., 2026; Jiang et al., 2026)。+Error LLM 条件消融了我们的智能体环设计：单次大语言模型调用接收每个错误轨迹并直接提出修复方案，但无法检查文件、查询真实值或迭代缩小根本原因。表 6 比较了 +Error（智能体，我们的方法）与 +Error LLM 在所有四种作者-模式组合下的表现。

在所有情景中，Agentic 分析在平均表现上均胜出。Agentic +Error 在所有四个情景中的加权平均表现均优于 +Error LLM，差距分别为 +12.2 pp (122B 深化)、+0.8 pp (122B 创造)、+3.2 pp (35B 深化) 和 +13.3 pp (35B 创造)。122B 创造情景是唯一接近平局的情况：两种条件得分均约为 27 pp，掩盖了其内部显著的发散（见下文）。

Agentic 错误分析能够生成更具迁移性的补丁。平均优势在分布内 SpreadsheetBench 和分布外 WikiTQ 列中均成立：仅提升大语言模型得分的修补方案在取得相近分布内得分时，往往导致 WikiTQ 和跨模型单元性能下降，而 Agentic 修补方案在两个维度上均保持正向优势。

为何 Agentic 环能生成更具可迁移性的补丁。对 33 个共享错误案例的定性研究证实了 \mathcal{A}^- 中的结构优势 (§2)：两条流水线仅在 4 个案例 (12.1%) 上达成强一致，而在 18 个案例 (54.5%) 中存在明显分歧。仅依赖执行日志的 LLM 分析器在解析错误信息出现的案例中，将解析失败过度归因为首要根本原因，占比达 57% (而 Agentic 为 14%)，且在至少一个案例中，为一条轨迹虚构出三个不同的故障原因，而产物评估确认该输出实际上已正确。通过访问产物并验证修复效果， \mathcal{A}^- 能够拒绝此类假阳性，并将每个补丁锚定于经验证的故障机制——从而生成跨模型规模和领域外情景可迁移的领域通用防护机制。

4.4. 可泛化的学成操作规程

我们检查由 122B Deepening + Combined 运行生成的 323 个地图块，以描述流水线提炼出的标准操作流程 (SoPs)。四种最普遍的主题共同涵盖了 546/323 个地图块引用（地图块可引用多个主题），并直接编码在主

SKILL.md 中；较少见的模式详见附录 A。

公式重新计算与写回验证 (178/323 个补丁)。每次编写公式后运行 `recalc.py`，并使用 `data_only=True` 重新打开以确认计算结果；跳过此步骤会导致单元保持未更新状态，这是运行中最为常见的错误模式。

工具选择：openpyxl 优于 pandas.to_excel() (177/323 个补丁)。使用 pandas 完成读取/变换逻辑，使用 openpyxl 完成写回操作；首先将输入文件复制到输出路径，以保留所有结构锚点。pandas.to_excel() 会静默破坏公式关系和命名值域。

显式回读验证 (138/323 个补丁)。写入完成后，重新打开输出文件，确认每个目标单元都包含期望值后再提交；错误的轨迹通常会特征性地省略此检查。

结构化编辑安全 (53/323 个补丁)。按降序删除行以防止索引偏移损坏；在编辑前复制输入的工作簿以保留格式和公式。错误轨迹记录了所有故障模式；成功轨迹验证了防护工作流的有效性。

小众特性被重定向到 references/。低支持的观测结果不会被丢弃，而是被引导至 13 个补充参考文件，而非主文件 SKILL.md。例如，单元颜色提取、在特定业务逻辑下的 FIFO 与 LIFO 不匹配等情况。这反映了既有的技能设计实践：程序指导从通用到特定案例逐步细化，主文档编码通用的工作流规则，而 references/ 则作为按需查询的层，用于处理罕见的边缘情况。Trace2Skill 通过轨迹证据自动恢复这一层级结构，无需人工整理。

5. 相关工作

智能体技能 Anthropic 的技能框架将技能形式化为轻量级、由专家撰写的文档，这些文档编码了针对特定任务领域的标准操作流程 (SOP)。这些技能设计用于动态加载、渐进式披露，并与多种智能体框架兼容 (Anthropic, 2026b)。SkillsBench (Li et al., 2026b) 提供了首个系统性的技能质量基准，揭示出虽然经过精心筛选的人工编写技能通常能提升性能，但完全依赖参数化知识自动生成的技能很少有帮助。此外，研究发现一小部分专注的技能始终优于单一的庞大文档。类似地，Li (2026) 证明，通过深度技能增强的单智能体能够达到多智能体框架的性能水平，尽管技能检索仍是主要瓶颈。SWE-Skills-Bench (Han et al., 2026) 在真实的软件工程任务中评估了技能注入效果，报告称当技能与任务上下文良好匹配时，平均取得 +1.2% 的性能提升，但在上下文不匹配时则出现显著性能下降。AgentSkillOS 与 SkillNet 进一步扩展了这一生态系统，涵盖技能选择与治理 (Li et al., 2026a; Liang et al., 2026)。我们的立场：我们基于已有的共识——高质量、聚焦的技能至关重要。然而，我们关注一个更狭窄且未被充分探索的问题：在给定一个范围有限的单一技能的前提下，系统性轨迹分析能将其改进多少？SWE-Skills-Bench 所揭示的对上下文不匹配的脆弱性直接促使我们采用归纳提炼可泛化的模式，而非过度拟合特定查询的设计选择。

智能体自我演化中的经验记忆。早期工作表明，对执行轨迹进行迭代反馈可以显著提升智能体的行为表现。例如，Voyager (Wang et al., 2023) 通过开放式交互积累可复用的技能，而 Reflexion (Shinn et al., 2023) 则通过对其过往成功与失败的言语自我反思来优化决策。在此基础上，后续研究聚焦于将轨迹衍生的洞见存储在检索库中，并在后续任务中查询这些信息以增强性能 (Ouyang et al., 2026; Fang et al., 2026; Wang et al., 2024; Qian et al., 2024; Nottingham et al., 2024; Liu et al., 2025)。我们的立场：尽管这些系统依赖于在测试时从情景记忆库中检索信息，我们则探索一种根本不同的方法：将经验提炼为静态、可共享的陈述性技能。这一差异由两个特性驱动。首先，跨多个轨迹的归纳压缩能够消除单个回合中的偶然性，从而生成稳健的原则而非局部的轶事。其次，提炼出的陈述性技能具有架构无关性，可在不同智能体和模型规模间无缝共享，而检索库通常与生成它们的具体框架紧密耦合。

自动技能自演化。我们的工作最接近的邻居能够从智能体轨迹中自动演化技能。SkillWeaver (Zheng et al., 2025) 通过结构化探索生成网络 API 技能。AutoSkill (Yang et al., 2026) 通过提取-维护-重用生命周期, 从用户聊天轨迹中在线创建和更新技能。XSkill (Jiang et al., 2026) 维护双重存储: 技能编码任务级标准操作流程, 经验编码上下文敏感的动作级指导。EvoSkill (Alzubi et al., 2026) 迭代诊断失败并验证技能更新, 是目前最接近的单一系统邻居。Memento-Skills (Zhou et al., 2026) 采用有状态的 Markdown 技能, 通过读写环增量更新。除了完全自动化的演化之外, Anthropic 的技能创建系统 (Anthropic, 2026a) 代表了人机协同优化的最新水平, 从业者基于小规模测试集上的智能体输出进行定性修正。

我们的定位: 尽管先前工作已积极探索基于轨迹的技能演化, Trace2Skill 引入了三个关键差异化特性。(1) **多对一合并:** 我们同时合并所有轨迹局部补丁, 而非按轨迹顺序逐个编辑技能, 避免了顺序依赖性和对早期观测的过拟合。(2) **全面的声明式产物:** 我们致力于统一的、类 Anthropic 风格的技能目录, 而非狭义的 API 对象、双存储或检索增强混合体。(3) **无需测试时检索:** 演化后的技能可直接消费, 使其天然兼容任意智能体流水线。此外, 尽管同期的自动化系统依赖专有大模型 (如 Claude), 我们的完整流水线仅使用参数量小至 35B 的开源模型即可实现稳健演化。最后, Trace2Skill 作为人工监督 (如 Anthropic 的技能创建系统) 的可扩展补充, 从数百条轨迹中提炼经验教训, 避免了人工审查带来的瓶颈。

技能与策略的协同演化。SkillRL (Xia et al., 2026a) 通过强化学习协同演化技能与模型策略, 将技能视为局部经验触发器 (“当 X 时, 执行 Y”), 而非完整的标准操作流程。类似地, ARISE 和 MetaClaw (Xia et al., 2026b; Li et al., 2026c) 探索了双时间尺度的在线自适应机制, 支持持续的策略更新。我们的立场: 与需要参数更新的协同演化方法不同, 我们严格研究冻结模型、无需训练、面向制品级别的适应, 确保所提炼的技能完全与模型无关。

6. 结论

我们提出了 Trace2Skill, 这是一种自动技能创建与适应的框架, 模拟人类专家构建技能的方式: 通过大量经验积累广泛领域知识, 再将其转化为简洁、声明式的成果。与逐条更新技能不同, Trace2Skill 通过并行部署一批分析子智能体, 从互不重叠的轨迹批量中提出针对性的修改补丁。随后, 它利用归纳推理和程序化冲突预防机制, 将所有提议同时整合为一个连贯的技能目录。

我们的研究发现, 从一个模型轨迹中提炼出的技能在不同模型规模之间以及针对分布外任务时均表现出极强的泛化能力。此外, 分析表明, 同时整合大量轨迹局部经验可显著提升计算效率和下游性能, 且单一可移植的技能文件夹优于基于检索的逐例经验。通过将输出结构化为分层目录 (例如, 在主 ‘SKILL.md’ 文件中包含通用原则, 而在 ‘references/’ 子目录中存放特定案例的启发式规则), Trace2Skill 能够有效编码既具泛化性的模式, 又包含细致入微的特定案例陷阱。

局限性与未来工作方向。作为一项正在进行的工作, 本文在以下方面存在局限性, 我们计划在不久的将来予以解决: (1) **编辑补丁的因果效应量化:** 目前补丁是整体合并的, 难以分离出任意单一修改的边际贡献或潜在干扰。我们旨在开发方法, 严格量化单个轨迹衍生补丁对最终技能的因果影响。(2) **追踪特定技能部分的效用:** 我们尚未实现一种机制, 以动态追踪智能体在推理过程中对生成的技能目录中特定部分的依赖程度。未来工作将聚焦于细粒度归因追踪, 以确定不同组件 (例如, 特定检查清单条目与参考文件) 的确切效用, 从而实现对无效或分散注意力的技能部分的自动化剪枝。

References

- Salaheddin Alzubi, Noah Provenzano, Jaydon Bingham, Weiyuan Chen, and Tu Vu. Evoskill: Automated skill discovery for multi-agent systems, 2026. URL <https://arxiv.org/abs/2603.02766>.
- Anthropic. How to create a skill with claude through conversation. Claude Tutorials, 2026a. URL <https://claude.com/resources/tutorials/how-to-create-a-skill-with-claude-through-conversation>.
- Anthropic. What are skills? Claude Help Center, 2026b. URL <https://support.claude.com/en/articles/12512176-what-are-skills>. Access Date: 2026-03-22.
- Runnan Fang, Yuan Liang, Xiaobin Wang, Jialong Wu, Shuofei Qiao, Pengjun Xie, Fei Huang, Huajun Chen, and Ningyu Zhang. Memp: Exploring agent procedural memory, 2026. URL <https://arxiv.org/abs/2508.06433>.
- Tingxu Han, Yi Zhang, Wei Song, Chunrong Fang, Zhenyu Chen, Youcheng Sun, and Lijie Hu. Swe-skills-bench: Do agent skills actually help in real-world software engineering?, 2026. URL <https://arxiv.org/abs/2603.15401>.
- Guanyu Jiang, Zhaochen Su, Xiaoye Qu, and Yi R. Fung. Xskill: Continual learning from experience and skills in multimodal agents, 2026. URL <https://arxiv.org/abs/2603.12056>.
- Kimi Team. Kimi introduces agent swarm: Let 100 ai agents work for you. Kimi Blog, 2026. URL <https://www.kimi.com/blog/agent-swarm>.
- Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedat-tention, 2023. URL <https://arxiv.org/abs/2309.06180>.
- Hao Li, Chunjiang Mu, Jianhao Chen, Siyue Ren, Zhiyao Cui, Yiqun Zhang, Lei Bai, and Shuyue Hu. Organiz-ing, orchestrating, and benchmarking agent skills at ecosystem scale, 2026a. URL <https://arxiv.org/abs/2603.02176>.
- Jiachun Li, Pengfei Cao, Zhuoran Jin, Yubo Chen, Kang Liu, and Jun Zhao. Mirage: Evaluating and explaining inductive reasoning process in language models, 2025. URL <https://arxiv.org/abs/2410.09542>.
- Xiangyi Li, Wenbo Chen, Yimin Liu, Shenghan Zheng, Xiaokun Chen, Yifeng He, Yubo Li, Bingran You, Haotian Shen, Jiankai Sun, Shuyi Wang, Binxu Li, Qunhong Zeng, Di Wang, Xuandong Zhao, Yuanli Wang, Roey Ben Chaim, Zonglin Di, Yipeng Gao, Junwei He, Yizhuo He, Liqiang Jing, Luyang Kong, Xin Lan, Jiachen Li, Songlin Li, Yijiang Li, Yueqian Lin, Xinyi Liu, Xuanqing Liu, Haoran Lyu, Ze Ma, Bowei Wang, Runhui Wang, Tianyu Wang, Wengao Ye, Yue Zhang, Hanwen Xing, Yiqi Xue, Steven Dillmann, and Han chung Lee. Skillsbench: Benchmarking how well agent skills work across diverse tasks, 2026b. URL <https://arxiv.org/abs/2602.12670>.
- Xiaoxiao Li. When single-agent with skills replace multi-agent systems and when they fail, 2026. URL <https://arxiv.org/abs/2601.04748>.
- Yu Li, Rui Miao, Zhengling Qi, and Tian Lan. Arise: Agent reasoning with intrinsic skill evolution in hierarchical reinforcement learning, 2026c. URL <https://arxiv.org/abs/2603.16060>.

- Yuan Liang, Ruobin Zhong, Haoming Xu, Chen Jiang, Yi Zhong, Runnan Fang, Jia-Chen Gu, Shumin Deng, Yunzhi Yao, Mengru Wang, Shuofei Qiao, Xin Xu, Tongtong Wu, Kun Wang, Yang Liu, Zhen Bi, Jungang Lou, Yuchen Eleanor Jiang, Hangcheng Zhu, Gang Yu, Haiwen Hong, Longtao Huang, Hui Xue, Chenxi Wang, Yijun Wang, Zifei Shan, Xi Chen, Zhaopeng Tu, Feiyu Xiong, Xin Xie, Peng Zhang, Zhengke Gui, Lei Liang, Jun Zhou, Chiyu Wu, Jin Shang, Yu Gong, Junyu Lin, Changliang Xu, Hongjie Deng, Wen Zhang, Keyan Ding, Qiang Zhang, Fei Huang, Ningyu Zhang, Jeff Z. Pan, Guilin Qi, Haofen Wang, and Huajun Chen. Skillnet: Create, evaluate, and connect ai skills, 2026. URL <https://arxiv.org/abs/2603.04448>.
- Brian S. Lin, Jiaxin Yuan, Zihan Zhou, Shouli Wang, Shuo Wang, Cunliang Kong, Qi Shi, Yuxuan Li, Liner Yang, Zhiyuan Liu, and Maosong Sun. On llm-based scientific inductive reasoning beyond equations, 2025. URL <https://arxiv.org/abs/2509.16226>.
- Yitao Liu, Chenglei Si, Karthik Narasimhan, and Shunyu Yao. Contextual experience replay for self-improvement of language agents, 2025. URL <https://arxiv.org/abs/2506.06698>.
- Zeyao Ma, Bohan Zhang, Jing Zhang, Jifan Yu, Xiaokang Zhang, Xiaohan Zhang, Sijia Luo, Xi Wang, and Jie Tang. Spreadsheetbench: Towards challenging real world spreadsheet manipulation, 2024. URL <https://arxiv.org/abs/2406.14991>.
- Minesh Mathew, Dimosthenis Karatzas, R Manmatha, and CV Jawahar. Docvqa: A dataset for vqa on document images. corr abs/2007.00398 (2020). *arXiv preprint arXiv:2007.00398*, 2020.
- Kolby Nottingham, Bodhisattwa Prasad Majumder, Bhavana Dalvi Mishra, Sameer Singh, Peter Clark, and Roy Fox. Skill set optimization: Reinforcing language model behavior via transferable skills, 2024. URL <https://arxiv.org/abs/2402.03244>.
- Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T. Le, Samira Daruki, Xiangru Tang, Vishy Tirumalashetty, George Lee, Mahsan Rofouei, Hangfei Lin, Jiawei Han, Chen-Yu Lee, and Tomas Pfister. Reasoningbank: Scaling agent self-evolving with reasoning memory, 2026. URL <https://arxiv.org/abs/2509.25140>.
- Panupong Pasupat and Percy Liang. Compositional semantic parsing on semi-structured tables, 2015. URL <https://arxiv.org/abs/1508.00305>.
- Cheng Qian, Shihao Liang, Yujia Qin, Yining Ye, Xin Cong, Yankai Lin, Yesai Wu, Zhiyuan Liu, and Maosong Sun. Investigate-consolidate-exploit: A general strategy for inter-task agent self-evolution, 2024. URL <https://arxiv.org/abs/2401.13996>.
- Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023. URL <https://arxiv.org/abs/2303.11366>.
- Qwen Team. Qwen3.5: Accelerating productivity with native multimodal agents, February 2026. URL <https://qwen.ai/blog?id=qwen3.5>.
- Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models, 2023. URL <https://arxiv.org/abs/2305.16291>.

- Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory, 2024. URL <https://arxiv.org/abs/2409.07429>.
- Peng Xia, Jianwen Chen, Hanyang Wang, Jiaqi Liu, Kaide Zeng, Yu Wang, Siwei Han, Yiyang Zhou, Xujiang Zhao, Haifeng Chen, Zeyu Zheng, Cihang Xie, and Huaxiu Yao. Skillrl: Evolving agents via recursive skill-augmented reinforcement learning, 2026a. URL <https://arxiv.org/abs/2602.08234>.
- Peng Xia, Jianwen Chen, Xinyu Yang, Haoqin Tu, Jiaqi Liu, Kaiwen Xiong, Siwei Han, Shi Qiu, Haonian Ji, Yuyin Zhou, Zeyu Zheng, Cihang Xie, and Huaxiu Yao. Metaclaw: Just talk – an agent that meta-learns and evolves in the wild, 2026b. URL <https://arxiv.org/abs/2603.17187>.
- Chenfei Xiong, Jingwei Ni, Yu Fan, Vilém Zouhar, Donya Rooein, Lorena Calvo-Bartolomé, Alexander Hoyle, Zhijing Jin, Mrinmaya Sachan, Markus Leippold, Dirk Hovy, Mennatallah El-Assady, and Elliott Ash. Co-DETECT: Collaborative discovery of edge cases in text classification. In Ivan Habernal, Peter Schulam, and Jörg Tiedemann (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 354–364, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-334-0. doi: 10.18653/v1/2025.emnlp-demos.25. URL <https://aclanthology.org/2025.emnlp-demos.25/>.
- Yutao Yang, Junsong Li, Qianjun Pan, Bihao Zhan, Yuxuan Cai, Lin Du, Jie Zhou, Kai Chen, Qin Chen, Xin Li, Bo Zhang, and Liang He. Autoskill: Experience-driven lifelong learning via skill self-evolution, 2026. URL <https://arxiv.org/abs/2603.01145>.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023. URL <https://arxiv.org/abs/2210.03629>.
- Boyuan Zheng, Michael Y. Fatemi, Xiaolong Jin, Zora Zhiruo Wang, Apurva Gandhi, Yueqi Song, Yu Gu, Jayanth Srinivasa, Gaowen Liu, Graham Neubig, and Yu Su. Skillweaver: Web agents can self-improve by discovering and honing skills, 2025. URL <https://arxiv.org/abs/2504.07079>.
- Huichi Zhou, Siyuan Guo, Anjie Liu, Zhongwei Yu, Ziqin Gong, Bowen Zhao, Zhixun Chen, Menglong Zhang, Yihang Chen, Jinsong Li, Runyu Yang, Qiangbin Liu, Xinlei Yu, Jianmin Zhou, Na Wang, Chunyang Sun, and Jun Wang. Memento-skills: Let agents design agents, 2026. URL <https://arxiv.org/abs/2603.18743>.

Table 1 | 主要结果以差异值 (%) 形式展示。技能作者 = 进化出该技能的模型 (行分组); 技能使用者 = 推理时使用的模型 (列分组)。参考行保持绝对得分以提供上下文。进化行显示带符号的差异值, 颜色强度对应各列; **green** = 表示提升, **red** = 表示下降。差异值: 深化对比 人工编写基准; 创造对比 参数化 (无技能) 基准。平均值: 对分布内 SpreadsheetBench (验证/软/硬, 两种模型规模) 和分布外 WikiTQ (两种模型规模) 进行等权重平均, 表示为相对于相应基准的差异值。

Condition	Skill User: Qwen3.5-122B-A10B				Skill User: Qwen3.5-35B-A3B				Avg↑
	SpreadsheetBench			OOD	SpreadsheetBench			OOD	
	Vrf↑	Soft↑	Hard↑	WikiTQ↑	Vrf↑	Soft↑	Hard↑	WikiTQ↑	
Reference (absolute scores)									
No Skill	27.67	28.90	17.57	21.50	19.00	18.00	4.60	13.33	18.35
Human-Written	48.33	36.30	17.03	74.68	9.67	13.03	3.37	9.02	31.57
Parametric	26.17	36.60	17.50	23.73	20.17	13.70	3.87	20.14	20.80
Skill Author: Qwen3.5-122B-A10B									
Deepening (init: Human-Written)									
+Error	+17.50	+10.30	+10.40	+1.62	+27.00	+9.44	+2.86	+9.26	+9.18
+Success	-21.83	-8.57	+0.04	-10.35	+9.16	+3.57	+1.56	+12.09	-0.90
+Combined	+21.50	+10.87	+12.50	+4.56	+21.16	+8.84	+1.80	+6.64	+9.19
Creation (init: Parametric)									
+Error	+22.83	+3.77	+5.87	+7.89	+8.66	+9.53	+4.00	+2.06	+7.04
+Success	+15.33	-0.93	+4.33	+23.70	+12.83	+11.57	+6.13	+30.36	+17.62
+Combined	+0.16	-9.23	-1.40	+32.32	-1.17	+3.73	+1.36	+29.70	+14.96
Skill Author: Qwen3.5-35B-A3B									
Deepening (init: Human-Written)									
+Error	+16.67	+8.50	+8.14	-6.36	+17.33	+9.17	+4.83	+2.71	+4.47
+Success	-22.00	-8.83	-0.50	+1.46	+11.00	+3.64	+0.83	+43.23	+9.85
+Combined	+6.67	+3.87	+4.17	+2.65	+20.00	+5.77	+2.36	+42.20	+14.78
Creation (init: Parametric)									
+Error	+1.00	-7.70	+1.03	+57.65	+3.83	+7.30	+2.66	+12.66	+18.26
+Success	+5.33	-4.57	+2.43	+9.09	+5.66	+5.80	+2.63	+3.31	+4.54
+Combined	-0.84	-9.17	-1.63	+30.82	-0.17	+4.40	+1.26	+18.00	+11.69

Table 4 | Parallel consolidation vs. sequential editing on SpreadsheetBench (same-model Deepening, +Error only, %). Seq-B: skill updated after every batch of B trajectories. **Bold** = best per column.

Condition	Skill User: 122B			Skill User: 35B			Time↓
	Vrf↑	Soft↑	Hard↑	Vrf↑	Soft↑	Hard↑	
Seq-B=4	59.00	40.63	20.63	26.17	22.37	7.47	~15 min
Seq-B=1	61.83	44.40	25.40	26.00	23.83	10.57	~60 min
Parallel (ours)	65.83	46.60	27.43	27.00	22.20	8.20	~3 min

Table 5 | Trace2Skill (Human Written+Combined) vs. ReasoningBank on SpreadsheetBench (same-model Deepening, %). ReasoningBank retrieves success and failure memories at inference via Qwen3-Embedding-8B; +Combined distills the same trajectory pool into a single portable skill with no retrieval module. **Bold** = best per column.

Condition	Skill User: 122B			Skill User: 35B		
	Vrf↑	Soft↑	Hard↑	Vrf↑	Soft↑	Hard↑
ReasoningBank (Ouyang et al., 2026)	56.00	40.10	21.30	20.50	17.30	4.97
Human-Written+Combined (ours)	69.83	47.17	29.53	29.67	18.80	5.73

Table 6 | Agentic error analysis (+Error) vs. single-LLM-call error analysis (+Error LLM) across all Author-Mode combinations (%). Same-model cells are shaded; plain cells are cross-model transfer. **Bold** = better of the two conditions per cell.

Condition	Skill User: Qwen3.5-122B-A10B				Skill User: Qwen3.5-35B-A3B				Avg	
	SpreadsheetBench			OOD	SpreadsheetBench			OOD		
	Vrf	Soft	Hard	WikiTQ	Vrf	Soft	Hard	WikiTQ		
Skill Author: Qwen3.5-122B-A10B										
Deepening										
+Error (ours)	65.83	46.60	27.43	76.30	36.67	22.47	6.23	18.28	40.75	
+Error LLM	67.00	43.93	25.23	39.81	25.00	22.43	6.23	11.24	28.58	
Creation										
+Error (ours)	49.00	40.37	23.37	31.62	28.83	23.23	7.87	22.20	27.84	
+Error LLM	27.17	27.73	16.20	47.26	19.83	17.60	4.70	23.30	27.08	
Skill Author: Qwen3.5-35B-A3B										
Deepening										
+Error (ours)	65.00	44.80	25.17	68.32	27.00	22.20	8.20	11.73	36.04	
+Error LLM	37.83	22.93	12.83	77.05	30.50	20.17	8.73	9.95	32.83	
Creation										
+Error (ours)	27.17	28.90	18.53	81.38	24.00	21.00	6.53	32.80	39.06	
+Error LLM	22.00	27.67	16.60	54.61	23.50	16.87	4.93	11.24	25.76	

A. 定性分析所得的次级研究问题

以下标准操作程序出现在 122B 深化 + 组合运行中，获得中等程度支持（10–15/323 个补丁），并在进化出的技能中编码，但未在正文部分讨论。

目标值域和答案位置验证（15/323 个补丁）。在编写之前，请从任务元数据中验证确切的目标工作表名称、单元格范围和 `answer_position` 字段。误读这些字段——写入错误的工作表或偏移一个单元格的范围——会导致静默失败，不会产生错误消息，但得分将为零。

数据类型和日期时间保留（15/323 个补丁）。将日期和数值值写为原生 Python 类型，而不是字符串。pandas 的日期解析和 openpyxl 的单元赋值都可能静默地将日期时间值转换为字符串；在写入前检查每一列的 `dtype`，并使用 openpyxl 的原生日期时间赋值。

编辑前的作业本结构探索（成功主导，~ 13/323 补丁）。列出所有工作表，检查行/列布局，并在任何写入操作前验证标题位置。这种预编辑探索可防止错误工作表和错误值域的失败，占本次运行中 151 个偏向成功的补丁的相当大一部分。

B. 提示模板与中间输出

本附录重现了每个流水线阶段使用的关键提示模板，并展示了具有代表性的中间输出，以使整个流水线完全透明且可复现。

B.1. 阶段 1：智能体系统提示模板

智能体 π_θ 在轨迹收集过程中遵循以下系统提示。在推理时，技能 S_0 会被添加到用户上下文中。需要注意的是，这与标准技能加载过程不同，在标准流程中，智能体最初仅能访问技能描述。我们通过将 SKILL.md 内容预加载到系统提示中来简化这一过程，因为 Trace2Skill 专注于改进一个固定的、相对于任务已知的目标技能。因此，无需像标准技能加载那样进行技能选择。重要的是，使用 Trace2Skill 技能的智能体仍然需要通过程序化方式发现由预加载的 SKILL.md 指向的资源（例如资源和脚本），这些内容并未被预加载。

Stage 1 — Agent System Prompt (abbreviated)

角色：你是一个专家 role（例如，电子表格分析）智能体。

技能背景：[此处插入 S_0 的内容]

任务：描述任务和输入文件。

可用工具：描述工具和环境。例如，bash（shell 执行）用于具有文件系统访问权限的 ReAct。

格式：ReAct 风格的交互——在推理迹和工具调用之间交替进行，直到任务完成。

B.2. 阶段 2：分析师提示模板及示例补丁

在第二阶段，补丁提出智能体首先类似于 (Ouyang et al., 2026) 一样抽取错误和成功记忆项，这些是可泛化的轨迹级知识，可能对未来的任务执行有帮助。随后，智能体读取原始技能目录，然后提出一个补丁，将记忆项编码到技能中。

B.2.1. 错误分析员提示 (\mathcal{A}^-)

Error Analyst System Prompt (abbreviated)

角色: 你是一个专注于 XXX 任务的故障分析智能体。

任务: 给定智能体的执行产物（日志 + 生成文件）和真实值，诊断 智能体失败的原因，识别 导致失败的根本原因，并通过实现一个最小修复方案来 验证你的诊断，使智能体的输出与真实值一致。你的分析必须是 **系统性的、基于证据的**，且 **可复现的**。当能够验证时，请勿猜测。

必需的工作流（强制要求）:

1. 理解任务和失效表面——确切地识别输出中哪里出了问题。
2. 追踪失败到智能体行为——定位导致不匹配的决策或代码步骤。
3. 通过最小修复验证根本原因——编写修复后的输出，并重新与真实值进行评估。
4. 重新评估——如果仍然失败，返回步骤 1 至 3 并修改你的诊断。

输出: 生成 (1) 故障原因项 — 基于可观察智能体行为的系统性、因果性原因；(2) 故障记忆项 (≤ 3) — 智能体应记住以避免类似故障的可泛化洞察。

B.2.2. 成功分析员提示 (\mathcal{A}^+)

Success Analyst System Prompt (abbreviated)

角色: 你是一位在人工智能智能体系统成功模式分析方面的专家。

任务: 给定一个成功的智能体轨迹，识别出有助于得出正确答案的可泛化行为模式。

要求:

广泛覆盖 — 轨迹中的每一种有效行为都必须被某种模式捕获。

频率感知 — 覆盖更多实例的模式应优先列出；罕见行为应被吸收至最近的更广泛模式中。

泛化性 — 每个模式必须描述一种通用的机制，而不是单一的任务特定细节。

输出: 一组紧凑的 成功记忆项，包含标题、描述，以及观察到的有效行为的具体示例。

B.2.3. 示例轨迹级补丁（错误分析器输出）

每位分析员为单一轨迹生成一个结构化补丁。以下示例来自一名错误分析员 (\mathcal{A}^-) 处理一条失败轨迹的情况，其中智能体在指定值域外删除了行。该补丁建议在行操作安全性方面新增一个 SKILL.md 小节；此补丁连同另外 52 个引用相关结构化编辑失败的补丁，后来被整合进“结构化编辑安全性”标准操作流程 (§4.4)。

Example Stage 2 Trajectory-Level Patch (error analyst, one trajectory)

分析师推理：单个错误记录揭示了一个清晰的失败模式：智能体删除了指定值域外的行，并且无法区分部分为空的数字行与完全为空的填充行。该技能在行/列操作或值域边界强制执行方面缺乏任何指导。

```
--- a/SKILL.md
+++ b/SKILL.md
@@ -55,0 +55,26 @@ ## Formula Verification Checklist

+### Row and Column Operations
+
+#### Range Boundary Enforcement
+- When a task specifies a cell range (e.g., F14:M83), operations
```

```

+ MUST stay within those exact boundaries
+- Parse range format: start_col + start_row : end_col + end_row
+- Never apply operations to rows/columns outside the range
+
+### Row Deletion Logic
+Before deleting any row, verify BOTH conditions:
+1. Target columns are empty (as specified by task)
+2. Row falls within the explicit deletion range
+
+**Do NOT delete** completely empty rows across ALL columns unless
+explicitly instructed -- these often serve as structural padding.
+
+### Post-Operation Validation
+After row/column operations:
+- [ ] Count remaining rows matches expected count
+- [ ] No header/footer/padding rows outside range were affected
+- [ ] Data integrity is maintained in adjacent columns
+- [ ] Range boundaries were respected (no off-by-one errors)

```

B.3. 第三阶段：合并操作符提示与示例整合补丁

B.3.1. 合并操作符提示 (M)

Merge Operator System Prompt (full)

您是一名技能编辑协调员。您会收到多个独立提出的补丁，每个补丁都建议对一个技能文件夹进行修改。您的任务是将其合并成一个连贯且无重复的补丁。

准则：

- 去重：**当多个补丁提出相同或非常相似的修改时，保留最佳版本（最具体、表达最准确）。
- 解决冲突：**如果补丁对同一部分内容提出了相互矛盾的修改，选择理由更充分的修改，或把两者合并成一个更优的修改。
- 保留独特见解：**不同的补丁解决不同的问题——包含所有独特的、非冗余的修改。
- 保持简洁性：**合并后的补丁应具有所有输入补丁中唯一编辑的 \leq 总和。去除冗余内容。
- 确保独立性：**合并补丁中的编辑必须是行级独立的——即使在不同的操作之间，也不得有两个编辑针对重叠的行或同一段文本。
- 原子性的创建/链接配对：**对 references/*.md 的 create 操作与插入指向它的链接的 SKILL.md 编辑是一个不可分割的配对——要么同时保留，要么同时删除。

普遍模式偏差：当多个补丁独立提出相似的修改以解决同一类失败或成功模式时，将这种循环视为任务具有系统性属性的证据。优先保留此类普遍修改，并将其表述为一般性原则，而非特定实例的修复。

B.3.2. 示例最终合并补丁 p^* （完全合并层级之后）

以下摘录展示了在 SpreadsheetBench-Verified 上对 323 个独立轨迹补丁进行四层分层合并后，最终合并补丁 p^* 产生的推理过程和代表性编辑。

Example Stage 3 Final Patch Output (excerpt)

```
{
  "reasoning": "Merged 3 patches addressing mixed failure/success
evidence. Key consolidation decisions: (1) Synthesized recalc.py
workflow from all patches using the most prominent CRITICAL WARNING
placement, CSV fallback validation, and a verification loop;
(2) Consolidated library selection guidance into a comprehensive
decision tree; (3) Combined row deletion guidance emphasizing
bottom-up/right-to-left deletion order from all patches;
(4) Merged formula validation checklists without redundancy.",
  "edits": [
    {
      "file": "SKILL.md",
      "op": "insert_after",
      "target_section": "# Requirements for Outputs",
      "content": "## Important Automation Guidelines\n\n
**Prefer Python over VBA for Automation**: When tasks request
VBA macros or spreadsheet automation, implement the logic in
Python using openpyxl/pandas instead. This provides better
error handling, easier debugging, cross-platform compatibility,
and avoids macro security issues."
    },
    {
      "file": "SKILL.md",
      "op": "insert_after",
      "target_section": "## Important Requirements",
      "content": "#### When to Compute Directly in Python\n
If LibreOffice/recalc.py cannot evaluate certain formula
patterns (complex array formulas, advanced INDEX/MATCH
combinations), compute the logic directly in Python and write
results as values. Use this fallback when formula returns errors
after recalculation despite correct syntax."
    }
  ],
  "changelog_entries": [
    "Added automation guidelines preferring Python over VBA",
    "Consolidated recalc.py warnings into a single CRITICAL WARNING"
  ]
}
```

B.3.3. 示例翻译差异：最终合并补丁 p^*

由第 3 阶段生成的 JSON 补丁 p^* 被程序化地转换为统一格式的差异文件，并直接应用于技能目录。以下摘录展示了在 SpreadsheetBench-Verified 上最终应用的差异文件中的两个片段，分别对应两种最常见的 SoPs：公式重新计算强制执行（178/323 个补丁）和工具选择指导（177/323 个补丁）。

Example Stage 3 Applied Diff (excerpt from final *p*')

```

--- a/SKILL.md
+++ b/SKILL.md
@@ -126,3 +261,18 @@ ## Common Workflow

## Common Workflow

+### CRITICAL WARNING: Formula Recalculation Is Mandatory
+
+**If you write ANY formulas to an Excel file using openpyxl,
+you MUST run recalc.py before considering the task complete.**
+
+Formulas written via openpyxl exist only as text strings until
+recalculated. Without running recalc.py:
+- Cells return None/empty when read with data_only=True
+- Evaluation fails even if formulas are syntactically correct
+- The output file is incomplete
+
+This is non-negotiable. Do not proceed to verification or
+delivery until recalc.py confirms success.
+
@@ -138,3 +285,9 @@ ### Working with openpyxl

+### Tool Selection Warning
+
+**CRITICAL**: When modifying spreadsheets that contain existing
+formulas you need to preserve:
+- Use openpyxl (load_workbook() then save()) -- formulas remain
+  as strings
+- Avoid pandas (to_excel()) -- silently converts formulas to
+  static values permanently

```