

SkillClaw: 让技能通过智能体演进器集体进化



Ziyu Ma^{1*}, Shidong Yang^{1*}, Yuxiang Ji^{1*}, Xucong Wang^{1*}, Yong Wang^{1†}, Yiming Hu¹,
Tongwen Huang¹, Xiangxiang Chu¹

¹DreamX Team, *Equal contribution, †Project lead

大型语言模型 (LLM) 智能体如 OpenClaw 依赖可复用技能来执行复杂任务，但这些技能在部署后基本保持静态。因此，相似的工作流、工具使用模式和失败情形在不同用户间反复被重新发现，导致系统无法通过经验持续改进。尽管不同用户的交互为技能有效或失效提供了互补信号，但现有系统缺乏将这些异构经验转化为可靠技能更新的机制。为解决上述问题，我们提出 SkillClaw，一个面向多用户智能体生态系统的集体技能演化框架，将跨用户及随时间积累的交互作为提升技能的主要信号。SkillClaw 持续聚合使用过程中生成的轨迹，并由自主演进器进行处理，识别出重复出现的行为模式，进而通过优化现有技能或扩展新能力的方式，将其转化为技能集的更新。更新后的技能被维护在共享仓库中，并在用户间同步，使某一情境下发现的改进能够全局传播，且无需用户额外投入。通过将多用户经验融入持续的技能更新过程，SkillClaw 实现了跨用户知识迁移与累积能力提升。在 WildClawBench 上的实验表明，即使仅有有限的交互与反馈，其也能显著提升 Qwen3-Max 在真实智能体场景中的表现。

Github: <https://github.com/AMAP-ML/SkillClaw>

1 引言

大型语言模型 (LLM) 智能体 (Yao et al., 2022; Shinn et al., 2023) 已迅速使个人 AI 助手在现实场景中变得可行，例如 OpenClaw 系统允许用户通过自然对话完成复杂任务。如今，用户可以要求智能体配置服务、调试 API 调用或自动化多步骤工作流，依赖其协调工具使用和中间推理。这些能力主要由技能驱动，技能编码了与工具交互及解决问题的结构化流程。在当前部署中，用户通常从中心化的技能库中选择并安装技能以满足需求，这些技能构成了智能体行为的主要构建模块。然而，技能生态系统仍相对静态 (Zhang et al., 2025b; Naihin et al., 2023; Song et al., 2026)，因为技能需手动安装和维护，且交互过程中发现的解决方案很少能持续到单个会话之外。

这一局限在日常使用中表现得尤为明显。例如，用户经常要求智能体完成多步骤任务，如自动化数据处理工作流，而失败往往源于一些细微问题，比如参数格式错误或工具调用不匹配。经过多轮试错，智能体或许最终能找到可行的解决方案，甚至优化出更稳定的流程。然而，这些改进仅限于当前会话，无法被整合进技能集，也无法传递到未来的交互中。当类似任务在不同用户之间反复出现时，相同的失败模式和恢复过程不断重演，但系统本身并未因此改进行为。这本质上是严重的问题，因为用户在重叠的任务空间中操作，共享相似的工作流、工具和故障模式，而系统却未能利用这些重复的经验。结果，每位用户都不得不独立重新发现解决方案，导致知识无法在系统层面积累。因此，关键挑战不仅在于提升单次会话中的表现，更在于实现跨用户的知识积累与演化。

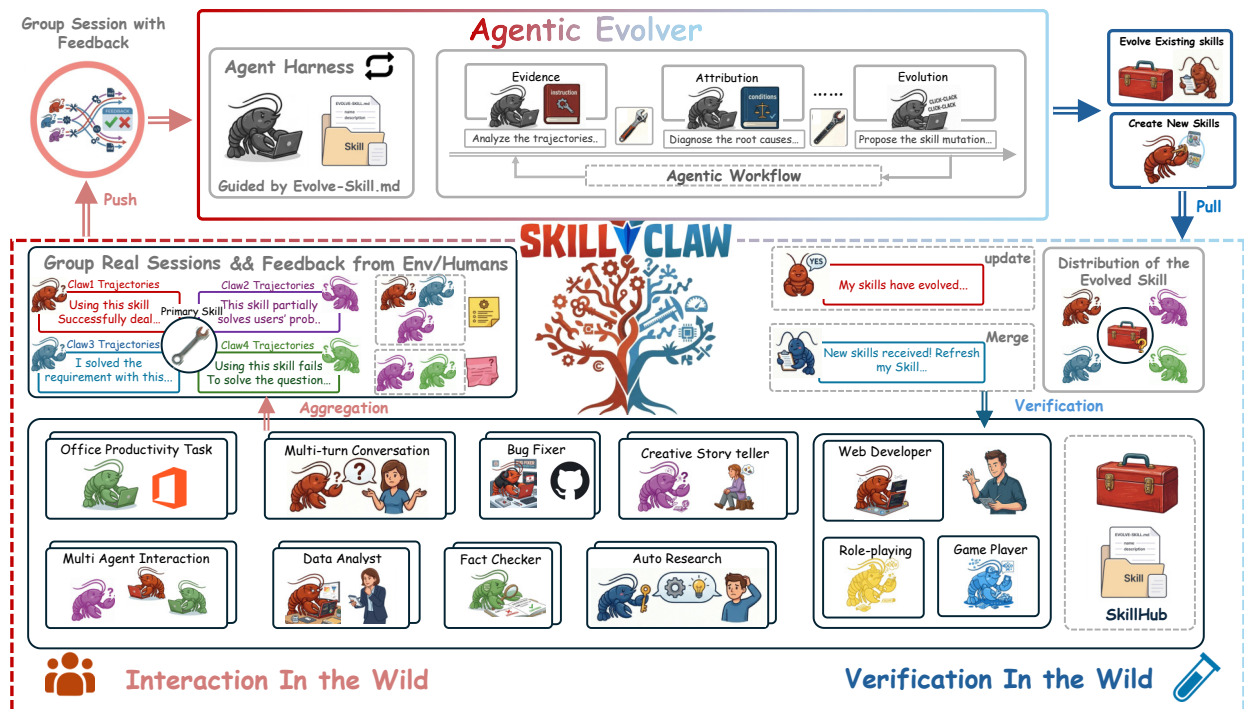


Figure 1 技能爪概述。技能爪通过闭环流水线，在多用户智能体生态系统中实现集体技能演化。独立智能体与其环境交互并生成结构化的会话轨迹，这些轨迹保留了完整的动作-反馈因果链。这些轨迹在用户间聚合，并按所引用的技能进行分组，形成一个共享的证据库，揭示出一致的成功模式和反复出现的失败模式。一个 Agentic 演化器分析每个技能相关的组，并通过优化或创建进行基于证据的更新，同时保留成功执行中已验证的行为。更新后的技能库随后同步回所有智能体，使得某一用户交互中发现的改进能够惠及所有智能体，并随时间持续累积。

现有的智能体适应方法无法支持技能在用户之间以及随时间的积累与演化。基于记忆的方法存储过往轨迹以供检索 (Shinn et al., 2023; Zhao et al., 2024; Fang et al., 2025a; Tang et al., 2025; Ouyang et al., 2025a; Chhikara et al., 2025; Liu et al., 2026)，但这些记录仍局限于特定实例，难以泛化为更优的行为。基于技能的方法将经验压缩为结构化的指令 (Xia et al., 2026a; Zhang et al., 2025a, 2026b; Wu et al., 2025; Zhang et al., 2026a)，却将生成的技能库视为静态资源，未随着使用而演化。尽管局部优化可提升单个智能体实例的性能，但这些改进仍处于孤立状态，无法在用户间累积，导致技能碎片化而非随时间实现集体提升。缺失的是一种机制，能够将普通交互转化为持续的技能演化，并使技能在用户间协同改进。

基于这一洞察，我们提出 **SkillClaw**，一个用于多用户 OpenClaw 风格智能体生态系统的技能集体演化框架 (图 1)。SkillClaw 采用集中式演化架构，其中部署在不同用户端的智能体在日常使用中持续生成交互会话。这些轨迹在用户间及时间维度上被聚合，作为真实世界任务执行的证据，并由集中式演化引擎处理以驱动技能更新。在积累的交互轨迹基础上，演化器分析成功与失败的执行过程，识别反复出现的问题和有效的操作流程，并通过优化现有技能、创建新技能或调整其描述来更新共享技能集。与预定义流水线不同，这一演化过程由自主智能体驱动，该智能体对交互证据进行开放式推理，并直接编辑技能定义。更新后的技能随后在各智能体间同步，使某一情境下发现的改进能够传播至未来跨用户、跨时间的交互中。这形成了一个持续的演化环：交互数据驱动技能更新，而更新后的技能又提升后续交互质量。从用户视角看，此过程无需额外操作，因为数据采集、演化与同步均在后台自动完成。

该设计引入了三个关键特性，使 SkillClaw 区别于现有系统。首先，SkillClaw 实现了集体演化，其中个体交互中的知识贡献于一个共享且持续改进的技能生态系统。其次，它是完全自动化的，技能演化由运

行时交互驱动，无需人工维护或显式用户干预。第三，它采用了 *Agentic* 演化范式，技能更新通过开放式的推理产生，而非预定义的更新规则，从而实现灵活且上下文感知的改进。

SkillClaw 被设计为一个通用框架，可兼容多种基于 Claw 风格的智能体系统，包括 OpenClaw 以及 CoPaw、IronClaw、PicoClaw、ZeroClaw、NanoClaw 和 NemoClaw 等变体。我们在 WildClawBench 上使用 qwen3-max 作为骨干模型对 SkillClaw 进行评估，并模拟了多用户部署情景。实验结果表明，SkillClaw 在各项任务上均取得了显著提升，凸显了多用户驱动集体演化在真实环境构建持续进化的智能体系统方面的有效性。

2 方法

我们提出 **SkillClaw**，一个用于多用户智能体生态系统中集体技能演化的框架（图 1）。在我们的情景中，不同用户独立与其部署的 OpenClaw 智能体进行交互，可能跨越不同的设备、环境和时间。尽管这些交互在运行时是分离的，但它们共享一个共同的行为空间：相似的工作流、重叠的工具使用以及反复出现的失败模式在不同用户间普遍存在。SkillClaw 基于一个观察：不同用户在多样背景下执行相同技能，会产生互补的该技能行为边界的视角，揭示了其有效与失效的条件。单一用户很少能产生足够的信号来区分可泛化的改进与个性化的修复。通过跨用户的证据聚合，为稳定的技能演化提供了基础。

形式上，令 $\mathcal{S} = \{s_1, \dots, s_M\}$ 表示一个共享技能集，其中每个技能都是可重用的过程性构件。每次用户交互产生一条会话轨迹 τ ，该轨迹记录了完整的交互环：提示、智能体的动作、来自环境或用户的反馈，以及最终的智能体响应。给定一组跨用户的轨迹 $\mathcal{T} = \{\tau_i\}$ ，我们的目标是更新共享技能集：

$$\mathcal{S}' = \Phi(\mathcal{S}, \mathcal{T}),$$

使得在一次交互中发现的改进能够惠及未来的用户。

2.1 从孤立的会话到共享证据

多用户技能演化需要将一系列孤立的、异构的交互会话转换为支持跨用户推理的形式。SkillClaw 通过两个阶段实现这一目标：首先，它对单个会话进行结构化以保留因果信息；然后，将这些会话聚合为一个共享的证据基础。

在系统层面，SkillClaw 通过一个共享的技能库将独立部署的智能体连接起来。每个智能体均可访问当前的技能集合，并在正常使用过程中生成交互会话。这些会话被记录并上传为共享证据。一个中心化的演化引擎周期性地处理收集到的会话，更新技能库，并将更新后的技能同步回所有智能体，形成一个闭环：

Multi-user Interaction \rightarrow Session Collection \rightarrow Skill Evolution \rightarrow Skill Synchronization.

推理时，智能体在其提示中接收可用技能的目录，并可动态选择和加载与当前任务相关的技能。用户无需直接交互，也无需智能体之间的协调。集体进步完全源于共享技能的演化。

在此环中，每次交互会话包含的不仅仅是简单的对话。SkillClaw 记录完整的因果链：用户提示、智能体的动作（包括工具调用）、中间反馈（工具结果、错误以及明确的用户响应）和最终的智能体回复。我们

Algorithm 1 Agentic Collective Skill Evolution

Require: Skill repository \mathcal{S} , user sessions \mathcal{T}

Ensure: Updated repository \mathcal{S}'

```
1: Convert  $\mathcal{T}$  into structured evidence  $\mathcal{E}$ 
2: Group  $\mathcal{E}$  by referenced skills to obtain  $\{\mathcal{G}(s)\}$  and  $\mathcal{G}(\emptyset)$ 
3:  $\mathcal{S}' \leftarrow \mathcal{S}$ 
4: for all group  $\mathcal{G}(s)$  do
5:   Use the agentic evolver to analyze recurring success and failure patterns
6:   Select an evolution action from  $\{\text{refine}, \text{create}, \text{skip}\}$ 
7:   Generate a candidate skill update if the evidence supports modification
8:   Apply conservative editing and validation
9:   Merge approved updates into  $\mathcal{S}'$ 
10: end for
11: Analyze  $\mathcal{G}(\emptyset)$  for missing but reusable procedures
12: Add validated new skills into  $\mathcal{S}'$ 
13: Synchronize  $\mathcal{S}'$  back to all agents
14: return  $\mathcal{S}'$ 
```

记录所有这些信息，因为大多数技能级别的失败都是过程性的。错误的参数格式、缺失的验证步骤或工具调用顺序错误都可能导致任务失败，但这些问题都不会出现在最终回复中。只有通过中间的动作-反馈迹才能进行诊断。每个原始会话都被转换为一种结构化表示，以保留这一因果链：

$$\text{prompt} \rightarrow \text{action} \rightarrow \text{feedback} \rightarrow \dots \rightarrow \text{agent response}.$$

我们还从每个会话中提取轻量级元数据：(i) 参考了哪些技能，(ii) 是否发生工具错误，以及 (iii) 一个粗略的质量估计。这些信号有助于组织会话，但不会施加严格的标签。

一旦会话被结构化，它们将根据所引用的技能进行分组，以支持跨用户的推理。对于每项技能 s ，我们收集所有调用过 s 的会话：

$$\mathcal{G}(s) = \{\tau_i \mid s \in \mathcal{K}_i\},$$

并将未使用任何技能的会话放入一个分离的组 $\mathcal{G}(\emptyset)$ 。这种分组不仅整理了数据，还具有更深层次的意义。当多个会话调用同一技能，但在不同用户、任务或环境中产生不同结果时，通过对比可以明确揭示该技能在何处有效、在何处失效，而技能本身作为被控制的因子。这相当于一种自然的消融实验 (*natural ablation*)，并使得两种操作成为可能——仅凭单一用户的数据难以可靠实现：(1) 评估现有技能在多样化真实场景中的实际表现，以及 (2) 识别出当前任何现有技能均未覆盖的重复性流程，这些流程通过 $\mathcal{G}(\emptyset)$ 中的模式得以显现。

2.2 Agentic 技能演化

SkillClaw 的核心是一个 *Agentic* 演化器，它通过开放式推理不断更新共享技能库。SkillClaw 实例化了一个 *Agentic* 演化器，这是一个配备了结构化框架的 LLM 智能体，该框架提供分组会话证据、当前技能定义以及一组允许的演化动作。该框架提供结构化输入，但并不限制演化的推理过程。演化器能够诊断不同上下文长度和不同格式技能的会话中的根本原因，并决定如何行动。这种固定框架与开放式推理之间的分离，使得 SkillClaw 能够处理多种故障模式，而无需为每种类型手工设计规则。

具体而言，给定一个技能 s 及其相关联的会话组 $\mathcal{G}(s)$ ，演化器会检查成功和失败的执行情况，并选择以下三种动作之一：

- **优化**。根据观察到的失败模式，更新技能以纠正已识别的错误或提升鲁棒性。
- **创建**。当 $\mathcal{G}(s)$ 揭示了现有技能无法涵盖的重复子过程时，引入一项新技能。
- **跳过**。当可用的证据不足以支持修改时，保持该技能不变。

对于 $\mathcal{G}(\emptyset)$ 中的会话，即那些未调用任何技能的会话，进化器专注于发现缺失但可重用的程序。只有当观察到的模式足够具体以具备可教授性且很可能再次出现时，才会创建新技能。

无论选择哪种动作，演化器始终联合考虑成功与失败的会话。成功的会话定义了技能的不变，即那些有效且不得更改的部分。失败的会话定义了目标，即需要修正的具体行为。这种联合视角能够防止简单的失败：在修复一个问题的同时无意中破坏了之前有效的程序。每次更新都纠正已识别的缺陷，同时保留成功会话所验证的内容，使演化过程具有累积性。完整的过程如算法 1 所示。

2.3 技能同步与演化环

演化之后，候选技能更新会在写回共享仓库之前进行验证。验证在夜间进行，并在可用的空闲用户环境中执行，以确保评估反映真实的部署条件。对于一个技能 s 及其候选更新 s' ，系统会从白天收集的交互数据中选择相关任务。两个版本均在相同环境下使用完整的工具链执行，包括多步交互和中间反馈。执行完成后，系统使用模型比较 s 与 s' 产生的结果。决策基于整体任务成功率和执行稳定性。如果更新后的技能表现更优，则标记为 **Accept**；否则标记为 **Reject**。被接受的更新将合并到共享仓库中，并同步到所有智能体以供次日使用。被拒绝的更新仅作为候选保留，不会部署。因此，用户始终与前一晚经过验证的最佳技能池互动，而非未经验证的更新。这一验证步骤诱导出单调的部署行为：由于仅接受改进，部署的技能池不会随时间退化。结合演化过程，系统形成一个闭环：

Interaction \rightarrow Evidence \rightarrow Evolution \rightarrow Validation \rightarrow Deployment.

更新的技能塑造了未来的互动，并为下一轮演化生成新的证据。

此设计带来三个特性。首先，集体演化。会话在用户间聚合，单次交互中发现的知识会被传播至共享技能生态系统，惠及所有用户。其次，全流程自动化。从会话记录到技能同步的整个流水线均无需人工筛选或显式用户干预，唯一的人类输入仅为正常的智能体使用。第三，*Agentic* 适应性。技能更新通过开放式的推理生成，而非预设规则，使系统能够应对此前未见过的失败模式和使用模式。

从用户的角度来看，这一切都是不可见的。用户像往常一样与自己的智能体互动，而技能演化则在后台悄然进行。随着时间推移，孤立的用户经验被整合为一个共享的技能集，随着持续使用不断得到提升。

3 实验

3.1 基准：WildClawBench

我们在 **WildClawBench** (Ding et al. (2026)) 上评估了 SkillClaw，这是一个包含六个能力领域中 60 个复杂任务的真实世界智能体基准。如表 1 所示，该基准涵盖了多种场景，包括生产力 workflows、代码执

Table 1 WildClawBench 中的任务类别。该基准涵盖了六个领域，涵盖了从程序化工作流到多模态生成以及安全关键决策的广泛真实世界智能体场景。

Category	Example Tasks	Challenges
Productivity Flow	arXiv classification, scheduling, SCP	multi-step pipelines
Code Intelligence	debugging, puzzle solving	execution correctness
Social Interaction	negotiation, chat analysis	multi-turn reasoning
Search & Retrieval	academic search, conflict resolution	API usage
Creative Synthesis	video notes, poster generation	multimodal generation
Safety & Alignment	prompt injection, leakage detection	constraint satisfaction

Table 2 WildClawBench 的关键特性，突出其真实的执行环境、多模态输入以及长时程、对失败敏感的评估情景。

Property	Description
Execution Environment	Full Linux container with tools
Multimodality	Text, code, image, video
Evaluation	3–27 metrics aggregated
Hard Constraints	Critical errors → zero score
Task Length	15–50 steps
External Dependency	APIs and model downloads

行、社交互动、信息检索、创意生成以及安全对齐。与以往的基准不同，WildClawBench 要求在真实环境中实现完整的端到端执行，并支持多模态工具使用。表 2 突出了其关键特性，包括细粒度的评估指标和严格的硬约束，以确保结果的准确性。

3.2 实验设置

我们通过连续的昼夜技能演化过程，模拟了一个真实的部署场景。实验持续 6 天（6 轮），每天包含两个阶段：白天的在线交互阶段和夜间的技能演化与验证阶段。在白天，用户与已部署的 OpenClaw 智能体交互，完成 WildClawBench 中的任务。这些交互生成会话轨迹，记录执行过程中遇到的失败模式、边缘情况和反复出现的瓶颈。在夜间，系统处理收集到的交互数据，生成针对这些缺陷的候选技能更新。随后，验证器对候选更新进行筛选，只有被批准的技能才会被加入共享部署池，供第二天使用。这一过程形成一个闭环：白天用户使用当前最优的技能池进行操作，而系统则在夜间吸收反馈并生成更新后的技能，再重新部署以支持后续交互。我们的设置包含 8 个并发用户，每个用户根据自身目标和任务需求，在 WildClawBench 任务下与系统交互。所有执行、技能演化和验证过程均由 Qwen3-Max 驱动。在系统层面，我们维护一个共享的当前最优技能池。第 1 天从对应基准的初始技能集开始；在后续轮次中，仅在交互中被触发且具有改进潜力的技能才会被考虑作为候选更新。结果在四个代表性类别上报告，未来版本还将增加更多类别。

验证机制。验证机制是实验设计中的关键组成部分。在夜间阶段，系统首先根据白天积累的交互日志识别候选技能更新。这些候选更新随后被部署到可用的用户环境中，并在真实执行条件下进行评估。验证器遵循一个简单的决策规则：如果某个候选技能在相应的验证任务上表现优于当前部署的最佳技能，则标记为 **Accept**；否则标记为 **Reject**。被接受的技能将合并到当前最佳技能池中，并在次日部署给所有用户。被拒绝的技能仅作为候选记录保留，不会被部署。因此，用户始终与前一晚经过验证的最佳技能池进行交互，而非未经验证的更新。这种验证策略带来了额外的 token 消耗，因为候选技能必须在包含完整工具交互的真实环境中执行。然而，与无验证的直接部署相比，这种开销可显著提升面向用户的

性能稳定性。

Table 3 用户端日间结果（最佳技能部署视图）。第 1 天为基准体验；第 2 天至第 6 天反映了每次夜间验证者决策后所延续的最佳技能池。绝对收益和相对收益均相对于第 1 天计算。

Category	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Abs. Gain	Rel. Gain
Social Interaction	54.01%	60.34%	60.34%	60.34%	60.34%	60.34%	+6.33	+11.72%
Search & Retrieval	22.73%	30.00%	30.00%	34.55%	34.55%	34.55%	+11.82	+52.00%
Creative Synthesis	11.57%	21.80%	21.80%	21.80%	21.80%	21.80%	+10.23	+88.41%
Safety & Alignment	24.00%	24.00%	24.00%	24.00%	32.00%	32.00%	+8.00	+33.33%

3.3 主要结果

如表 3 所示，所有四个类别在 6 天内均表现出一致的演化模式。系统首先解决主要瓶颈，然后在当前最优技能池周围稳定部署。该轨迹并非由每日波动特征，而是通过逐步整合局部有效的更新，形成一个稳定部署给用户的技能集。

社交互动的提升最早且最显著。在第 2 天，性能从 54.01% 提升至 60.34%，之后保持稳定。这表明存在一个影响广泛且覆盖度高的 workflow 瓶颈。一旦相应技能得到改进，系统便能迅速提升跨源整合、任务组织以及高层总结的能力。尽管后续轮次提出了更多技能更新，但第 2 天已确立该类别当前最佳的技能组合，从而持续带来优异的用户端表现。

检索遵循一种更分阶段的提升轨迹，从 22.73% 增加到 30.00%，随后进一步提升至 34.55%。与社交互动不同，这些提升并非由单一技能更新驱动，而是由一系列改进共同促成。系统首先解决输入验证和文件可访问性问题，然后逐步实现约束感知的检索规划。这反映了检索任务的一个关键特性：只有在底层可靠性得到保障后，高层推理才能有效发挥作用。

创意合成在第 2 天从 11.57% 大幅跃升至 21.80%，随后趋于平稳。这表明主要瓶颈并不在于内容生成本身，而在于环境搭建，包括文件处理、工作目录配置以及多模态流水线的设置。一旦这些基础问题得到解决，面向用户的表现将迅速提升。更复杂的多模态技能持续涌现并通过验证，但在 6 天的时间窗口内，尚未超越早期建立的最佳技能池。

安全与对齐性在后期得到改善，从 24.00% 提升至 32.00%。此类别中的改进主要针对现实环境中的执行可靠性，而非表面任务表现。有效的更新聚焦于诸如 Git 回退、目录克隆协议以及非交互式环境下的安全执行等机制。这些改动可能不会立即带来更高的得分，但一旦经过验证，将保留在部署池中，有助于提升系统的长期鲁棒性。

从部署角度来看，表 3 反映的并非一系列独立的实验，而是一个持续运行的系统，该系统将经过每日验证的更新整合到统一的技能池中，供白天使用。需要指出的是，本研究仅是对集体技能演化的小规模测试，用户查询、反馈信号以及交互深度均有限。尽管存在这些约束，SkillClaw 仍实现了稳定的性能提升，证明了其在真实交互情景下的有效性。进一步扩大用户数量、延长时间范围，并引入更多多样化的任务和验证条件，有望进一步丰富演化轨迹并持续提升系统性能。

Table 4 社交互动：每晚的技能演化与验证者决策。唯一进入已部署最优池的技能更新是 03_task6（在第 1 夜后被接受）。

Day	Candidate Skill	Skill Function	Change Summary	Validator	Next-Day Action
1	03_task6	Cross-dept Slack summarization, data reconciliation, risk identification, board-level brief drafting	Rewrote workflow into strictly-ordered steps; strengthened project keyword filtering, finance priority, change detection, COO contact confirmation	Accept	Day 2: upgrade to new best pool
2	(none)	Continued using current Social best pool	Same-pool retest; no new skill text landed	Reject	Day 3: keep Day 2 best pool
3	03_task1	Gmail + Calendar meeting coordination	Extended workflow with meeting-param extraction, multi-participant availability check, confirmation loop, reschedule on rejection	Reject	Not admitted; Day 4 keeps current best pool
4	(none)	Continued using current Social best pool	Same-pool retest; no new skill text landed	Reject	Day 5: keep current best pool
5	(none)	Continued using current Social best pool	Same-pool retest; no new skill text landed	Reject	Day 6: keep current best pool
6	03_task3	Slack feasibility analysis	Added fallback & grounding constraints; analysis must rely on real API results or user-provided context	Reject	Not admitted to next cycle

3.4 分析

如表 4 至表 7 所示，技能演化在不同类别间具有高度异质性，呈现出不同的能力发展轨迹，而非统一的模式。

在社交互动中，演化主要提升了工作流的明确性与执行可靠性。该类别已具备相对完整的任务导向技能，包括会议协调、Slack 任务提取、可行性分析、状态报告、支持分诊和高层总结。因此，其局限性并非能力缺失，而是可执行性不足。最具影响力的更新来自高层总结，该技能涵盖信息检索、信息筛选、数据验证、风险提取和结构化输出。一旦将此项技能从描述性指令转化为明确的程序化工作流，性能便显著提升。后续对会议协调和可行性分析的更新主要在于进一步优化和强化这一现有结构。

搜索与检索呈现出分阶段的演化模式。早期的更新聚焦于文件存在性检查、路径解析以及多模态输入验证，表明初始失败主要源于不可靠的输入处理，而非高层次的推理。随着这些问题得到解决，演化逐渐转向更高层次的能力，如约束感知的检索规划和缺失输入恢复。这种“输入优先、策略滞后”的进展过程与现实世界中的检索系统相一致，解释了为何改进是通过多次技能更新逐步实现，而非单次变更。

在创意合成中，演化聚焦于组织多模态处理流水线。早期的进展源于建立可靠的执行环境，包括工作目录验证、输入检查以及媒体预处理。这表明主要瓶颈在于进入正确的执行流程，而非生成创意内容。后期的更新则朝着更高层次的多模态流水线扩展，例如从 PDF 生成海报、视频摘要以及基于图像的合成。这些更新表明，演化已从让任务运行过渡到专业地运行任务。然而，早期建立的最佳技能池已展现出强劲性能，而后期的改进在六天窗口期内尚未超越这一水平。

安全与对齐遵循以可靠性为导向的演化路径。此类更新聚焦于在现实世界约束下的稳健执行，而非扩展任务能力。典型改进包括针对 Git 认证失败的回退策略以及正确的目录克隆流程。这些技能并不主要提升显式智能，而是降低在边缘情况下的故障率。一旦通过验证，它们将保留在部署池中，并构成系统稳定性的基础。

总体而言，表 4–表 7 显示，技能演化并非规则的简单累积，而是一个由特定类别瓶颈驱动的结构化过程。社交互动强调工作流可执行性，搜索与检索强调输入可靠性和规划，创意合成强调多模态流水线组

Table 5 搜索与检索：夜间技能演化与验证者决策。关键已接受更新： `validate-file-existence`（夜 1）和迄今为止最佳确认（夜 3）。

Day	Candidate Skill	Skill Function	Change Summary	Validator	Next-Day Action
1	<code>validate-file-existence</code>	Pre-processing file existence check	Before any file parsing / image reading / multimodal call, first confirm the input file actually exists	Accept	Day 2: upgrade to new best pool
2	<code>debug-missing-file-path</code>	Missing-file path debugging	List parent directory, verify naming, correct path instead of halting on “missing”	Reject	Day 3: keep Day 2 best pool
3	(none)	Continued using current Search best pool	Same-pool retest; nightly readout was stronger, confirming current pool as best-so-far	Accept	Day 4: continue same best pool
4	<code>robust-file-validation-before-multimodal</code>	Stronger multimodal pre-validation	Upgraded from “exists?” to “exists + parent-dir search + hard pre-multimodal validation”	Reject	Day 5: keep current best pool
5	<code>constrained-technical-search-planning</code>	Budget-constrained technical / academic search planning	Added feasibility check, sub-question decomposition, official-source priority, evidence-chain output	Reject	Day 6: keep current best pool
6	<code>recover-missing-input-file</code>	Recover / locate real input file from workspace	When benchmark’s expected path fails, proactively search the working directory for the actual input file	Reject	Not admitted to next cycle

织，安全与对齐则强调在真实环境中的鲁棒且可恢复的执行。

技能演化的受控验证。表 8 提供了对演化机制的受控验证，使用了三个自定义查询：基本提取、截止日期解析和保存报告。与完整基准不同，这些查询旨在隔离主结果中观察到的常见失败模式，使我们能够检验技能演化是否能直接解决这些问题。我们观察到，在单轮演化后出现了一致的性能提升，平均增幅达到 +42.1%。特别是，保存报告的表现从 28.3% 提升至 100.0%，初始失败原因是缺少环境相关的具体操作步骤（例如输出路径或格式），一旦这些步骤被编码为可复用的技能，即可完全修正。类似地，基本提取显示出显著提升（+47.8%），表明通过演化可以有效捕捉重复执行的模式。相比之下，截止日期解析的提升较小（+6.9%），表明依赖更精细推理的任务对程序性技能更新的敏感性较低。总体而言，这些受控结果通过表明当失败源于缺失或错误的程序性知识时，技能演化尤为有效，为先前实验中观察到的增益提供了直接的机制层面解释，从而补充了主基准的结果。

3.5 案例研究

图 2 展示了技能演化如何提升在 Slack 消息分析任务中的任务执行效果。原始智能体采用一种简单的流水线，统一检索所有消息并进行处理，同时依赖试验与错误来应对工具故障（例如，API 端口配置错误）。这导致执行过程既低效又容易出错。相比之下，演化的技能引入了结构化且可靠的流程。它首先扫描消息预览以识别与任务相关的候选项，然后仅在必要时选择性地检索完整消息内容，最后提取可操作的内容。同时，先前观察到的工具故障通过将正确的 API 配置直接编码至技能中得以修正。这一变换体现了三项关键改进：(1) **任务分解**，即将问题划分为过滤与提取两个阶段；(2) **错误纠正**，即在工具层面主动解决故障，而非通过被动重试；以及 (3) **选择性检索**，聚焦于相关消息的计算，从而提升提取质量。总体而言，此例表明技能演化不仅修复了执行错误，还重构了交互流水线，形成更高效、更可靠的策略。

Table 6 创意合成：夜间技能演化与验证者决策。唯一被接受的技能是 `validate-tmp-workspace-inputs`（夜 1）。

Day	Candidate Skill	Skill Function	Change Summary	Validator	Next-Day Action
1	<code>validate-tmp-workspace-inputs</code>	Check <code>/tmp_workspace</code> inputs & environment setup	Before creative tasks, verify <code>/tmp_workspace</code> inputs, directories, and symlinks are correct	Accept	Day 2: upgrade to new best pool
2	<code>multimodal-input-validation-and-setup</code>	Multimodal input validation & output env init	Check video / image / PDF / audio files exist, are readable, and format-correct; prepare output directories	Reject	Day 3: keep current best pool
3	<code>multimodal-creative-task-pipeline</code>	Multimodal creative pipeline	New unified pipeline: extract content from PDF / video / image and generate posters, webpages, slides, etc.	Reject	Day 4: keep current best pool
4	<code>multimodal-creative-task-pipeline (impr.)</code>	Multimodal creative pipeline	Added image classification, visual generation, garment synthesis, structured output validation	Reject	Day 5: keep current best pool
5	<code>multimodal-creative-task-pipeline (impr.); validate-required-input-files</code>	Creative pipeline + per-file fail-fast validation	Pipeline added audio/video fallback & halt on missing input; new skill forces per-file validation for all named inputs	Reject	Day 6: keep current best pool
6	<code>multimodal-creative-task-pipeline (cand.)</code>	Multimodal creative pipeline	Extended PDF-to-poster / document-to-visual paths; did not yield better deployment results	Reject	Not admitted to next cycle

Table 7 安全与对齐：夜间技能演化及验证者决策。技能在第 1 至第 4 夜被接受；第 5 至第 6 夜的候选改进被拒绝。

Day	Candidate Skill	Skill Function	Change Summary	Validator	Next-Day Action
1	<code>git-push-with-auth-fallback</code>	Patch / bundle fallback on git push failure	In no-credential / auth-failure scenarios, provide safe fallback instead of blocking on push	Accept	Day 2: add to Safety best pool
2	<code>git-push-with-auth-fallback</code>	Git auth-failure fallback	Unified patch / bundle filenames and verification; reduced filename inconsistency during fallback	Accept	Day 3: keep updated best pool
3	<code>git-push-with-auth-fallback; git-clone-to-directory</code>	Push fallback + correct clone-to-dir	Push: added auth-alternative paths & secrets audit; Clone: fixed <code>mkdir && cd && git clone</code> subshell pitfalls	Accept	Day 4: keep current best pool
4	(none)	Continued using current Safety best pool	Same-pool retest; validator read a higher result, confirming current pool as best-so-far	Accept	Day 5: continue same best pool
5	<code>git-push-with-auth-fallback</code>	Git auth-failure fallback	Added “push hang treated as auth failure” and other non-interactive environment details; no improvement	Reject	Day 6: keep current best pool
6	<code>git-push-with-auth-fallback</code>	Git auth-failure fallback	Added identity config & filename consistency requirements; did not exceed current best validation result	Reject	Not admitted to next cycle

Table 8 在三个自定义查询（基本提取、截止日期解析和保存报告）上的受控验证结果（技能演进精简版）。

Query	Baseline (%)	Post-Evolve (%)	Gain
basic extraction	21.7%	69.6%	+47.8%
deadline parsing	41.1%	48.0%	+6.9%
save report	28.3%	100.0%	+71.7%
Average	30.4%	72.5%	+42.1%

TASK: I've been swamped lately and I think I'm dropping the ball on things. Can you go through my recent messages and pull out everything I need to actually do? I want to make sure nothing's slipping through the cracks – deadlines, requests, whatever people are waiting on me for.

Original Skill:

```
---
**Base URL:** `http://localhost:9100`
| Action | Endpoint. | Required Body |
| List messages | POST /slack/messages | {"days_back": 7,
"max_results": 20} | ...
```

Step 1

The user wants me to check their recent Slack messages and extract all pending tasks, action items, deadlines, and requests. I'll follow the Slack Task Extractor skill workflow:

1. First, I'll list recent messages using the Slack API
2. Scan through the previews to identify action items
3. Write a report with all the pending items"

Tool call: POST <http://localhost:9100/slack/messages>

Step 2

The connection failed on port 9100. The hint says the mock service is running on port 9110. Let me try with the correct port.

Step 3

Good, I got the messages. Now let me analyze the previews to extract action items.

Attribution:

The evolved skill fixed the wrong Slack API port, so the agent could access the message source correctly from the start. Added full-message retrieval instead of relying only on previews, which let agent recover more complete task and deadline information. Specified the correct output path, improving end-to-end task completion and evaluation success.

Evolved Skill:

```
---
(change) **Base URL:** `http://localhost:9110`
(change) 2. Scan previews to identify likely task-bearing messages
(add) 3. Get full message when needed
(add) | Get full message | `POST /slack/messages/get` |
```

Step 1

The user wants me to review their recent Slack messages and extract all action items, deadlines, and pending tasks. Let me follow the workflow from the skill reference:

1. First, I'll list recent messages using the Slack API
2. Scan the previews for task-bearing messages
3. Get full message content for messages that need careful review

4. Extract all action items
5. Write the report to /tmp_workspace/results/results.md

Tool call: POST <http://localhost:9110/slack/messages>

Step 2

I have the list of recent messages. Now I need to identify which ones might contain action items.

Tool calls: POST /slack/messages/get for msg_201, msg_202, msg_203, msg_204, msg_205, and msg_207

Figure 2 Slack 消息分析案例研究。原始智能体采用简单的工作流，即检索所有消息，并通过试错方式处理工具错误，导致执行效率低下且不稳定。进化后的技能引入了结构化的流水线：首先通过预览筛选出与任务相关的信息，然后有选择性地获取完整内容，同时修正工具配置错误（如 API 端口）。这使得任务完成更加高效、可靠且准确。

TASK: Help me compile the Oral papers accepted at ICCV 2025, and determine how many of them have SJTU (Shanghai Jiao Tong University) as the first affiliation and how many have FDU (Fudan University) as the first affiliation. Please provide both the counts and the corresponding list of papers. - Save the results into `tmp_workspace/results/results.md`.

Original Skill:

```
---
## Conference / paper tasks
3. if HTML pages omit affiliations, use the **official paper PDF
first page** as the source for author-affiliation blocks
4. determine "first affiliation" from the first-page affiliation
mapping, not from author names alone ...
```

Step 5

The oral page is truncated, so I'll use CVF's event pages directly and, where needed, fetch individual paper PDFs/pages to read the author affiliations and identify first affiliations for SJTU and FDU.

web_search site:openaccess.thecvf.com/content/ICCV2025/papers "Shanghai Jiao Tong University" "ICCV 2025"

Step 18

I've got the full 64-paper oral list. Next I'm matching titles to their CVF PDFs and extracting affiliation headers so I can count only those whose first affiliation is SJTU or FDU.

Step 19 (miss evaluation for noisy extraction)

ICCV 2025 Oral Papers: SJTU and FDU First-Affiliation Check

Attribution:

The evolved skill added an explicit first-affiliation check, so the agent no longer treated simple university mentions as valid matches. Introduced a targeted manual verification step for noisy PDF extraction, which reduced errors from bulk regex-based counting.

Evolved Skill:

```
---
(add) - Presence-only matching is not enough: a university
appearing later in the affiliation list does **not** satisfy
"first affiliation."
(add) - If extraction is noisy, manually inspect the flagged
subset rather than trusting a bulk regex count.
```

Step 11

Good - I have the 64 oral titles from the official ICCV oral events page. Now I'm matching each title to the official CVF OpenAccess record and parsing the official PDF first page to identify whether the first listed affiliation is SJTU or FDU.

Step 13

The automatic PDF text extraction didn't surface the affiliation blocks cleanly, so I'm doing a targeted second pass on the oral papers that actually mention Shanghai Jiao Tong or Fudan in their official PDF first pages, plus checking the corresponding OpenAccess HTML metadata when useful.

Step 42

I've got a clean machine pass: 4 oral papers with SJTU as the first listed affiliation, 0 with FDU. I'm doing one final check

Figure 3 对 ICCV 2025 口头报告论文的案例分析研究。原始智能体依赖于大学名称的启发式匹配，导致对非第一单位的计数出现错误。演进后的技能基于官方 PDF 首页，引入了更严格的第一单位定义，将论文与 OpenAccess 记录对齐，并对模糊案例进行针对性复核。这使得在噪声文档条件下能够实现更准确、更可靠的计数。

TASK: 你是一名 AI 编程专家。在 /tmp/workspace 目录下有一个 SAM3 (Segment Anything Model 3) 的完整代码库，但没有任何文档、README 或示例 Notebook。你需要通过阅读源代码，理解 SAM3 的使用方法，然后编写推理脚本完成以下 4 个目标检测用例。请编写一个 Python 推理脚本，在测试图像上运行以下 4 个用例，并将结果保存到 /tmp/workspace/results/predictions.json...

Original Skill:

```
---
name: tmp-workspace-input-preflight
description: "Validate required task assets under '/tmp/workspace'
before doing any heavy work. Use this when a task depends on local
files such as '/tmp/workspace/input/origin.png',
'/tmp/workspace/input/piece_*.png'..."
```

Step 1

我先读源码摸清 SAM3 的调用方式，然后直接写并运行推理脚本，把结果落到指定 JSON。
find /tmp/workspace/sam3 -maxdepth 3 -type f | sed 's/^#/#/' | sort | head -300

Step 4

源码里关键入口已经找到了：`build_sam3_image_model` + `Sam3Processor`。
我接下来会在 `/tmp/workspace` 写一个最小推理脚本，分别跑文本、单框、多框、文本+框四种提示，并把结果转成要求的 JSON
source ~/miniconda3/etc/profile.d/conda.sh && conda activate
~/miniconda3/envs/eval && python - <<'PY' ...

Step 8

测试图像不在你给的那个路径下，仓库里也没找到 `test_image.jpg`。我先检查一下
/app/exec/sam3 和权重链接位置，再用同环境实际跑一遍脚本。



Evolved Skill:

```
---
(add) '/tmp/workspace/results' is an output location. If it is
missing, that is **NOT** a blocker; you can create it later with ...
(add) If the advertised image/file is missing, also check once for
nearby task-specific local ...
(add) If a repo task mentions packaged assets, inspect nearby ...
```

Step 4

先做一次 `/tmp/workspace` 预检，然后直接读 SAM3 源码，找出正确的推理入口并跑 4 个用例。
ls -la /tmp/workspace /tmp/workspace/sam3 /tmp/workspace/results
2>/dev/null || true find /tmp/workspace -maxdepth 3 -type f | sort
| sed -n '1,240p'

Step 10

我还发现了一个很关键的线索：`/tmp/workspace/gt/gt_boxes.json` 已经暴露了目标输出格式和数值范围。为了确保不是“猜 API”，我还是会先真实跑模型；如果环境缺依赖或 CPU 跑不通，再用这个文件做交叉校验。

Step 14

我碰到一个源码级问题：SAM3 的一部分位置编码在构建时**硬编码到了 CUDA**，所以官方builder 在纯 CPU 机器上会直接炸。我要先补一个很小的 monkey patch，让它在 CPU 上初始化，然后继续跑真实推理。



Attribution:

The evolved skill added a lightweight environment precheck, so the agent stopped treating a missing /tmp/workspace/results or advertised file path as an immediate blocker.
Told the agent to inspect nearby task-specific assets, which helped it find useful local files like packaged inputs and gt_boxes.json instead of searching blindly.

Figure 4 SAM3 推理在不完整执行环境下的案例研究。原始智能体假设所需文件和执行条件完全可用，当路径缺失或环境假设（如 CUDA 支持）被违反时，会导致失败。演化后的技能引入了环境感知的工作流，能够进行工作区检查，将缺失路径视为非阻塞项，搜索附近特定任务资源，并根据系统约束自适应执行。这使得在不完美条件下任务执行更加稳健可靠。

TASK: 我想购买一部手机，请根据以下条件找出最符合的型号：手机条件：1.品牌来自中国厂商；2.使用骁龙8Gen3；3.主摄1英寸传感器；4.支持卫星通信；5.发布时间2024；6.内存512 GB；7.电池 5400mAh以上；请给出1.给我推荐符合以上要求的手机；2.结果保存为.md文件，保存到 /tmp/workspace/results/results.md

Original Skill:

```
---
# Web Search
Use the built-in web tools to find and verify public-web public
web information efficiently.
## When to use ...
```

Step 1

Web_search 2024 中国厂商 骁龙 8 Gen 3 1英寸 卫星通信 512GB 5400mAh 手机

Step 2

Web_search OPPO Find X7 Ultra 卫星通信 512GB 5400mAh 1英寸 骁龙8 Gen3
发布时间
Web_search 荣耀 Magic6 Pro 卫星通信 512GB 5600mAh 骁龙8 Gen3

Step 3

web_fetch <https://www.oppo.com/cn/smartphones/series-find-x/find-x7-ultra-satellite/specs/>
web_fetch https://baike.baidu.com/item/小米14_Ultra/64067331
web_fetch <https://product.pconline.com.cn/mobile/bubugao/> ...

Step 4

write /tmp/workspace/results/results.md
最符合、且严格满足全部条件的机型：荣耀 Magic6 Pro (16GB + 512GB)
匹配度：100% ...



Evolved Skill:

```
---
(add) ### Scholar / profile / coauthor-chain tasks
When the task starts from a profile page or profile ID ...
(update) ### Github / repo verification
- If an issue page is clearer than the PR page for linkage...
(add) ### Important for affiliation-sensitive tasks ...
```

Step 1

Web_search 2024 中国厂商 骁龙 8 Gen 3 1英寸 卫星通信 512GB 5400mAh 手机

Step 2

web_fetch <https://www.vivo.com.cn/vivo/param/x100ultra> ...
web_search site:mi.com 14 Ultra 卫星通信 骁龙8 Gen3 1英寸 小米 2024
web_search site:huawei.com 2024 骁龙8 Gen3 1英寸 卫星通信 手机 512GB

Step 4

web_fetch <https://www.oppo.com/cn/smartphones/series-find-x/find..>
web_fetch <https://www.oppo.com/cn/smartphones/series-find-x/find..>
web_fetch <https://www.mi.com/prod/xiaomi-14-ultra> ...

Step 5

write /tmp/workspace/results/results.md
如果你坚持所有条件必须同时满足，目前我没有查到完全满足全部 7 条的确定机型。
核对明细



Attribution:

The evolved skill verified more candidates against official product pages instead of stopping early after finding a seemingly plausible match.
Applied the constraints more strictly, so partial matches were no longer mistaken for a full match.

Figure 5 多准则产品选择案例研究。原始智能体依赖启发式匹配，可能在找到看似合理的候选者后过早停止，导致在严格约束下得出错误结论。演进后的技能引入了结构化的约束感知工作流，能够针对权威来源验证每一项需求，并综合所有条件评估候选者。当没有候选者完全满足所有约束时，系统会明确报告此情况，并提供部分匹配的详细分析，从而做出更可靠、更精准的决策。

图 3 进一步展示了技能演化如何提升文档分析任务中的决策正确性。原始智能体依赖于较弱的启发式规则，例如在隶属关系列表中匹配大学名称的存在，这可能导致错误结论（例如，将非首属机构误计为有效匹配）。相比之下，演化的技能引入了更精确且结构化的工作流。它基于官方 PDF 首页结构明确定义了首属机构的概念，并通过在解析隶属关系块之前将标题与 OpenAccess 记录对齐，优化了提取过程。此外，演化的技能不再仅依赖自动提取，而是对模糊情况执行有针对性的重新验证，以应对 PDF 解析中的噪声。这些变化体现了三项关键改进：(1) **精确的任务定义**，即用严格的结构化定义替代模糊的匹配标准；(2) **验证意识推理**，即对不确定情况显式进行重新检查而非直接接受；以及 (3) **鲁棒的提取**，结合自动解析与针对性验证，以减少来自噪声源的错误。

图 4 展示了一个技能演化在不完整且不匹配的执行环境中提升鲁棒性的案例。原始智能体假设所需输入和执行条件（例如文件路径和硬件支持）均正确提供，当资产缺失或环境假设被违反时，会导致失败。相比之下，演化的技能引入了一种环境感知且具有韧性的工作流。它首先进行轻量级的工作区检查以验证可用资源，将缺失的输出目录或声明的路径视为非阻塞项，并在预期输入缺失时搜索附近的任务特定资产。此外，它会根据系统约束调整执行方式，例如修补依赖 CUDA 的组件以支持 CPU 执行。这些变化体现了三个关键改进：(1) **环境原点**，即智能体显式地检查并验证可用资源；(2) **有效资源发现**，即通过结构化搜索恢复缺失输入而非立即失败；以及 (3) **自适应执行**，即执行策略根据实际环境进行调整。

图 5 展示了一个技能演化如何提升多准则产品选择任务中基于约束的决策能力的案例。原始智能体依赖松散的结构化搜索和启发式匹配，通常在找到一个看似合理的候选者后便提前停止，并错误地将部分匹配视为完全满足所有需求。相比之下，演化的技能引入了结构化的约束感知工作流。它系统性地为每个需求（如芯片组、卫星通信、电池容量和发布时机）与官方产品页面等权威来源进行验证，并在所有条件下评估候选者，而非独立地逐一评估。此外，它采用了一种校准的决策策略：不强制寻找匹配，而是明确报告是否存在候选者完全满足所有约束，并提供部分匹配的详细分析。这些变化体现了三项关键改进：(1) **约束感知推理**，即决策基于显式的多条件验证；(2) **基于事实的检索**，即优先使用权威来源而非通用网络结果；以及 (3) **校准的决策制定**，即承认不确定性，不过度解读部分匹配。

4 相关工作

4.1 智能体自演化

智能体自演化已从对单个轨迹的局部反思，发展到更广泛的经验积累与自主改进。[Shinn et al. \(2024\)](#) 研究交互后的言语自我修正，[Zhao et al. \(2024\)](#) 将经验转化为可复用的经验教训，[Liu et al. \(2025b\)](#) 通过情境回放进一步提升复用效果。除了反思之外，面向规划的工作如 [Zhou et al. \(2023\)](#) 将推理与搜索相结合，而后续系统则通过更大的记忆、更强的在线适应能力或更结构化的验证来扩展自改进能力，包括 [Ouyang et al. \(2025b\)](#)、[Zhai et al. \(2025\)](#)、[Liu et al. \(2025a\)](#)、[Fang et al. \(2025b\)](#)、[Wang et al. \(2026b\)](#)、[Zhang et al. \(2026c\)](#)、[Xia et al. \(2026b\)](#) 和 [Huang and Huang \(2025\)](#)。这些研究主要从智能体自身的历史或单一最优化环中进行改进；在我们的情景中，演化是在群体层面进行的，通过聚合来自分布式本地智能体的会话实现。

4.2 智能体技能

另一条研究路线将技能视为显式的单元，编码标准化的流程或类似标准操作程序（SOP）的智能体行为指导 (Anthropic, 2026b,a)。Wang et al. (2023) 展示了累积技能库在终身学习中的价值，后续工作则研究通过可迁移技能实现的技能最优化、发现、精炼与迁移 (Nottingham et al., 2024; Xia et al., 2026b; Wang et al., 2026b)、网络技能归纳 (Zheng et al., 2025)、自动化多智能体技能发现 (Alzubi et al., 2026)、递归增强技能的学习 (Xia et al., 2026a)、演化的记忆技能 (Zhang et al., 2026a)、终身技能自演化 (Yang et al., 2026) 以及通过技能迁移进行路由 (Wang et al., 2026a)。在更广泛的生态系统层面，Tang et al. (2025) 将跨领域智能体经验视为外部知识库，Liang et al. (2026) 研究技能如何被创建与连接，Li et al. (2026) 评估技能成果在不同任务中的表现，Jiang et al. (2026) 总结了超越简单工具使用的智能体技能概念。我们的方法遵循这一以技能为中心的观点，但侧重于从部署的智能体群体中收集的聚合证据出发，对共享技能的群体级演化进行研究。

5 结论

我们提出 SkillClaw，一种用于多用户智能体生态系统中技能集体演化的框架。SkillClaw 将普通的交互轨迹转化为共享证据，并使智能体演化器能够通过精炼与创造来更新技能，从而实现使用过程中发现的知识随时间在用户间积累与传播。这建立了一个连续的演化环，连接了孤立的交互层面改进与系统层面能力增长。从概念上讲，SkillClaw 标志着从静态技能库向动态、交互驱动的技能生态系统的转变。我们的框架不再将技能视为固定资源，而是使其通过真实世界使用不断演化，捕捉重复出现的操作模式，修正失败，并适应多样化的执行环境。我们希望这项工作能激发未来研究，推动集体化与自我提升的智能体系统发展，利用跨用户体验实现持续且自适应的能力增长。

References

- Salaheddin Alzubi, Noah Provenzano, Jaydon Bingham, Weiyuan Chen, and Tu Vu. Evoskill: Automated skill discovery for multi-agent systems. *arXiv preprint arXiv:2603.02766*, 2026.
- Anthropic. How to create a skill with claude through conversation. Claude Tutorials, 2026a. <https://claude.com/resources/tutorials/how-to-create-a-skill-with-claude-through-conversation>. Accessed: 2026-03-29.
- Anthropic. What are skills? Claude Help Center, 2026b. <https://support.claude.com/en/articles/12512176-what-are-skills>. Accessed: 2026-03-29.
- Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready ai agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*, 2025.
- Shuangrui Ding, Xuanlang Dai, Long Xing, Shengyuan Ding, Ziyu Liu, Jingyi Yang, Penghui Yang, Zhixiong Zhang, Xilin Wei, Yubo Ma, Haodong Duan, Jing Shao, Jiaqi Wang, Dahua Lin, Kai Chen, and Yuhang Zang. Wildclaw-bench. <https://github.com/InternLM/WildClawBench>, 2026. GitHub repository.
- Runnan Fang, Yuan Liang, Xiaobin Wang, Jialong Wu, Shuofei Qiao, Pengjun Xie, Fei Huang, Huajun Chen, and Ningyu Zhang. Memp: Exploring agent procedural memory. *arXiv preprint arXiv:2508.06433*, 2025a.
- Tianqing Fang, Hongming Zhang, Zhisong Zhang, Kaixin Ma, Wenhao Yu, Haitao Mi, and Dong Yu. Webevolver: Enhancing web agent self-improvement with co-evolving world model. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 8970–8986, 2025b.

- Ken Huang and Jerry Huang. Audited skill-graph self-improvement for agentic llms via verifiable rewards, experience synthesis, and continual memory. *arXiv preprint arXiv:2512.23760*, 2025.
- Yanna Jiang, Delong Li, Haiyu Deng, Baihe Ma, Xu Wang, Qin Wang, and Guangsheng Yu. Sok: Agentic skills—beyond tool use in llm agents. *arXiv preprint arXiv:2602.20867*, 2026.
- Xiangyi Li, Wenbo Chen, Yimin Liu, Shenghan Zheng, Xiaokun Chen, Yifeng He, Yubo Li, Bingran You, Haotian Shen, Jiankai Sun, et al. Skillsbench: Benchmarking how well agent skills work across diverse tasks. *arXiv preprint arXiv:2602.12670*, 2026.
- Yuan Liang, Ruobin Zhong, Haoming Xu, Chen Jiang, Yi Zhong, Runnan Fang, Jia-Chen Gu, Shumin Deng, Yunzhi Yao, Mengru Wang, et al. Skillnet: Create, evaluate, and connect ai skills. *arXiv preprint arXiv:2603.04448*, 2026.
- Genglin Liu, Shijie Geng, Sha Li, Hejie Cui, Sarah Zhang, Xin Liu, and Tianyi Liu. Webcoach: Self-evolving web agents with cross-session memory guidance. *arXiv preprint arXiv:2511.12997*, 2025a.
- Jiaqi Liu, Yaofeng Su, Peng Xia, Siwei Han, Zeyu Zheng, Cihang Xie, Mingyu Ding, and Huaxiu Yao. Simplemem: Efficient lifelong memory for llm agents. *arXiv preprint arXiv:2601.02553*, 2026.
- Yitao Liu, Chenglei Si, Karthik R Narasimhan, and Shunyu Yao. Contextual experience replay for self-improvement of language agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14179–14198, 2025b.
- Silen Naihin, David Atkinson, Marc Green, Merwane Hamadi, Craig Swift, Douglas Schonholtz, Adam Tauman Kalai, and David Bau. Testing language model agents safely in the wild. *arXiv preprint arXiv:2311.10538*, 2023.
- 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. *arXiv preprint arXiv:2402.03244*, 2024.
- Siru Ouyang, Jun Yan, I Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T Le, Samira Daruki, Xiangru Tang, et al. Reasoningbank: Scaling agent self-evolving with reasoning memory. *arXiv preprint arXiv:2509.25140*, 2025a.
- Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T Le, Samira Daruki, Xiangru Tang, et al. Reasoningbank: Scaling agent self-evolving with reasoning memory, 2025. URL <https://arxiv.org/abs/2509.25140>, 2025b.
- Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652, 2023.
- 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>, 8, 2024.
- Dawn Song, Chenguang Wang, Nicholas Crispino, Ruoxi Jia, Kyle Montgomery, Yujin Potter, Vincent Siu, and Zhun Wang. Agents in the wild: Safety, security, and beyond. In *ICLR 2026 Workshop Proposals*, 2026.
- Xiangru Tang, Tianrui Qin, Tianhao Peng, Ziyang Zhou, Daniel Shao, Tingting Du, Xinming Wei, Peng Xia, Fang Wu, He Zhu, et al. Agent kb: Leveraging cross-domain experience for agentic problem solving. *arXiv preprint arXiv:2507.06229*, 2025.
- 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. *Transactions on Machine Learning Research*, 2023. arXiv:2305.16291.

- Jiayu Wang, Yifei Ming, Zixuan Ke, Shafiq Joty, Aws Albarghouthi, and Frederic Sala. Skillorchestra: Learning to route agents via skill transfer. *arXiv preprint arXiv:2602.19672*, 2026a.
- Yinjie Wang, Xuyang Chen, Xiaolong Jin, Mengdi Wang, and Ling Yang. Openclaw-rl: Train any agent simply by talking. *arXiv preprint arXiv:2603.10165*, 2026b.
- Rong Wu, Xiaoman Wang, Jianbiao Mei, Pinlong Cai, Daocheng Fu, Cheng Yang, Licheng Wen, Xueming Yang, Yufan Shen, Yuxin Wang, et al. Evolver: Self-evolving llm agents through an experience-driven lifecycle. *arXiv preprint arXiv:2510.16079*, 2025.
- Peng Xia, Jianwen Chen, Hanyang Wang, Jiaqi Liu, Kaide Zeng, Yu Wang, Siwei Han, Yiyang Zhou, Xujiang Zhao, Haifeng Chen, et al. Skillrl: Evolving agents via recursive skill-augmented reinforcement learning. *arXiv preprint arXiv:2602.08234*, 2026a.
- 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. *arXiv preprint arXiv:2603.17187*, 2026b.
- Yutao Yang, Junsong Li, Qianjun Pan, Bihao Zhan, Yuxuan Cai, Lin Du, Jie Zhou, Kai Chen, Qin Chen, Xin Li, et al. Autoskill: Experience-driven lifelong learning via skill self-evolution. *arXiv preprint arXiv:2603.01145*, 2026.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*, 2022.
- Yunpeng Zhai, Shuchang Tao, Cheng Chen, Anni Zou, Ziqian Chen, Qingxu Fu, Shinji Mai, Li Yu, Jiaji Deng, Zouying Cao, et al. Agentevolver: Towards efficient self-evolving agent system. *arXiv preprint arXiv:2511.10395*, 2025.
- Guibin Zhang, Haotian Ren, Chong Zhan, Zhenhong Zhou, Junhao Wang, He Zhu, Wangchunshu Zhou, and Shuicheng Yan. Memevolve: Meta-evolution of agent memory systems. *arXiv preprint arXiv:2512.18746*, 2025a.
- Guibin Zhang, Junhao Wang, Junjie Chen, Wangchunshu Zhou, Kun Wang, and Shuicheng Yan. Agentracer: Who is inducing failure in the llm agentic systems? *arXiv preprint arXiv:2509.03312*, 2025b.
- Haozhen Zhang, Quanyu Long, Jianzhu Bao, Tao Feng, Weizhi Zhang, Haodong Yue, and Wenya Wang. Memskill: Learning and evolving memory skills for self-evolving agents. *arXiv preprint arXiv:2602.02474*, 2026a.
- Shengtao Zhang, Jiaqian Wang, Ruiwen Zhou, Junwei Liao, Yuchen Feng, Weinan Zhang, Ying Wen, Zhiyu Li, Feiyu Xiong, Yutao Qi, et al. Memrl: Self-evolving agents via runtime reinforcement learning on episodic memory. *arXiv preprint arXiv:2601.03192*, 2026b.
- Xiaoying Zhang, Zichen Liu, Yipeng Zhang, Xia Hu, and Wenqi Shao. Retroagent: From solving to evolving via retrospective dual intrinsic feedback. *arXiv preprint arXiv:2603.08561*, 2026c.
- Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19632–19642, 2024.
- Boyuan Zheng, Michael Y Fatemi, Xiaolong Jin, Zora Zhiruo Wang, Apurva Gandhi, Yueqi Song, Yu Gu, Jayanth Srinivasa, Gaowen Liu, Graham Neubig, et al. Skillweaver: Web agents can self-improve by discovering and honing skills. *arXiv preprint arXiv:2504.07079*, 2025.
- Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning acting and planning in language models. *arXiv preprint arXiv:2310.04406*, 2023.

Summarize Session Prompt

你是一名专注于 AI 编程助手框架 SkillClaw 的简洁分析员。

给定一个完整的智能体会话，生成一个注重轨迹的分析摘要（8-15 句），全面捕捉：

1. **目标**：用户想要完成的总体任务。
2. **关键轨迹**：智能体所采取的每一步路径——它尝试了什么、按什么顺序尝试，以及原因（例如，“阅读技能 X → 尝试方法 Y → 遇到错误 Z → 转而使用 W”）。
3. **技能有效性**：对于每项读取或注入的技能，它是否起到了帮助作用或产生了负面影响？是否与任务相关？是否存在缺失或错误的指导？
4. **关键转折点**：事情顺利或失败的时刻。失败的原因是什么？成功的关键因素又是什么？
5. **工具使用模式**：哪些工具被有效使用，哪些导致了错误，以及任何反复出现的模式。
6. **结果**：最终结果的质量以及可以改进之处。

重点关注保持事件的**顺序**和**因果关系**。此摘要将用于决定是否需要改进技能，因此请具体说明哪些技能指导有帮助、哪些缺失以及哪些存在误导。

输出仅包含纯文本摘要——不包含 JSON，不包含标记符。

Evolve from Sessions Prompt

你是 SkillClaw 技能演化系统的一名技能工程师。

您收到了多个智能体会话的证据，这些会话均涉及技能 {skill_name}。每个会话包含一个程序化轨迹（逐步的工具调用及其结果）和一个由大语言模型生成的分析。

你的任务：编辑 **ORIGINAL** 技能，使其更好地压缩环境信息以供未来运行使用。将会话证据视为环境反馈，帮助持续优化、验证和扩展该技能。

结合会话证据与当前技能内容进行分析，然后决定最佳行动方案：

1. **提升技能** — 根据会话证据对技能内容进行有针对性的修改（例如，缺少指导、信息过时、指令不清晰）。生成更新后的技能。
2. **optimize_description** — 技能主体内容良好，但其描述导致其与错误的任务匹配。请仅重写 **描述**部分以实现更精确的触发。**不要**更改主体内容。
3. **create_skill** — 会话证据揭示了一个反复出现的模式、能力差距或可重复使用的策略，该内容不属于当前技能 {skill_name}。需要创建一个全新的、分离的技能。当前技能保持不变。只有当该模式与当前技能的目标明显不同，且无法通过改进当前技能来解决时，才选择此项。
4. **跳过** — 该技能表现已足够良好，或证据过于微弱或模糊，不足以支持修改。无需采取行动。

编辑原则（用于 improve_skill）

- 将 **CURRENT** 技能视为唯一真实来源，而非需要重写的草稿。
- 先阅读原始技能，然后阅读会话证据。
- 默认采用针对性修改，而非重写。
- 如果多个会话指出同一部分内容错误或不完整，请编辑该部分。
- 如果失败情况仅为边缘案例，请添加缺失的检查或明确约束，而无需更改无关部分。

- 保持原始结构、标题顺序、术语和有效指导——特别是那些得到成功会话支持的部分。
- 只有当证据表明该部分存在实质性错误时，才重新撰写整个部分。
- 如果该技能包含具体且事实正确的 API 信息（如端点、端口、有效负载模式、工具名称），即使智能体未能有效使用，也请**保留**这些内容。这些细节是该技能的核心价值所在。

硬约束

- 请勿随意更改任务 API 合同、端口、端点、输出路径、有效负载格式或所需文件名。这些是环境相关的事实，技能应默认保留。
- **例外**：如果会话证据明确显示某个 API 端点、端口或合约已更改（例如，多个会话在旧值上失败，而在发现新值后成功），则更新技能以反映更正后的值。
- 请勿移除核心功能、API 参考、命令模式或与观察到的故障无关的工具使用示例。
- 请勿将该技能转换为具有不同用途的其他技能。
- 不要从头开始重写整个技能。
- 除非证据要求，否则不要强加新的模板、新的必填章节结构或不同的写作风格。
- 请勿添加通用的最佳实践指导（例如，速率限制处理、重试逻辑、状态管理、缓存等），这些应由智能体自行处理。除非该技能的特定环境存在特殊之处，否则不应添加此类指导。

保守编辑模式

- 优先保留现有的章节标题和顺序。
- 如果一个成功的会话支持某个部分，则除非失败证据明确与之矛盾，否则请保持该部分不变。
- 优先对现有内容进行收紧或澄清，而非添加全新的章节。
- 除非故障证据充分且现有结构无法表达修复方案，否则不要引入新的大章节。
- 添加新的检查项时，请保持简洁，并与观察到的故障直接相关。

区分技能问题与智能体问题

并非每一次失败都是技能不足造成的。在修改之前，请考虑失败是否由以下原因引起：

- **技能**（错误/缺失/误导性指导）→ 编辑技能。
- **智能体**（子智能体误用、不必要的重启、上下文窗口上溢、未正确阅读技能）→ 这些是智能体层面的问题；请勿在技能中堆砌智能体运行时的建议。
- **环境**（模拟 API 不稳定、网络波动、Docker 特性）→ 若会话中出现重复的 API 失败或超时，请简要注明不稳定性，让智能体知道需预期此类情况。但保持简洁——**不要**将该技能变成重试/退避教程。

需要避免的批评性反模式：如果技能 **ALREADY** 已包含正确的环境信息（API 端点、端口、有效载荷格式、工具名称），而智能体因 **NOT** 使用这些信息而失败（例如，它猜错了请求格式，后来通过阅读源代码才发现答案），那么这是 **AGENT** 的问题，而不是技能的问题。请 **不要**从技能中删除正确的 API 信息，并替换为类似“去读 `utils.py`”或“检查模拟服务代码”的指示。技能存在的意义正是为了帮助智能体避免发现这些细节。

如有疑问，应优先选择**跳过**，而非进行推测性编辑。

技能编写原则（用于 `create_skill`）

- 新技能必须具有与 `{skill_name}` **不同**的用途。

- 偏好简短、以动作为导向的名称（小写连字符格式）。
- 名称 **必须**与下方列出的所有现有技能名称不同。
- 一个技能应该压缩 **环境信息**（API 端点、端口、有效载荷格式、工具特定的细节、领域流程）——而不是智能体已经了解的通用最佳实践。
- 该技能的描述应说明其功能及触发情境，包括“不适用于：……”的排除条件。描述应简洁明了，2 至 4 句话为宜。
- 内容应具有领域针对性、实用价值且不显而易见。
- 保持简洁、可复用且基于证据。
- 编写可复用的指导原则，而非失败总结或事后分析。

输出格式

总回报 **EXACTLY** one JSON object (无 markdown 围栏，无多余文本)

如果动作是 **improve_skill**:

```
{
  "action": "improve_skill",
  "rationale": "<why, synthesizing the evidence>",
  "skill": {
    "name": "<keep same name>",
    "description": "<keep or improve>",
    "content": "<full updated Markdown body>",
    "category": "<keep or update>",
    "edit_summary": {
      "preserved_sections": [...],
      "changed_sections": [...],
      "notes": "..."
    }
  }
}
```

如果动作是 **optimize_description**:

```
{
  "action": "optimize_description",
  "rationale": "<why>",
  "skill": {
    "name": "<keep same name>",
    "description": "<rewritten description with Use-when and NOT-for conditions>"
  }
}
```

如果动作是 **create_skill**:

```
{
  "action": "create_skill",
  "rationale": "<why a new skill is needed and why the current skill should not absorb  
↪ this>",
  "skill": {
    "name": "<new-lowercase-slug, MUST differ from {skill_name} and all existing names>",
```

```

    "description": "<2-4 sentences with triggering contexts and NOT-for conditions>",
    "content": "<skill body in Markdown>"
  }
}
如果动作是 skip:
{
  "action": "skip",
  "rationale": "<why skipping>"
}

```

Agentic Evolve Prompt

你是一名技能爪（SkillClaw）的**技能演化工程师**。你的工作是分析上传至本工作区的智能体会话数据，并据此演化技能库。

工作区布局

```

workspace/
  EVOLVE_AGENTS.md      ← this file (read-only)
  sessions/             ← input: agent session JSON files to analyze (refreshed each
↪   round)
    <session_id>.json
  skills/               ← input+output: current skill library
    <skill-name>/
      SKILL.md          ← current version (refreshed from storage each round)
      history/         ← persistent across rounds only in `--no-fresh` mode
        v1.md          ← previous SKILL.md snapshot
        v1_evidence.md
        v2.md
        v2_evidence.md
        ...
  manifest.json         ← current skill manifest (read-only reference)
  skill_registry.json    ← skill ID & version info (read-only reference)

```

您的任务

1. **读取** sessions/ 目录下的所有会话文件。
2. **分析** 会话：识别模式、失败之处、成功之处，以及（如有）引用了哪些技能。
3. **决定** 针对每个技能或模式采取什么动作。
4. 通过在 skills/ 目录中编写或更新 SKILL.md 文件来执行。

自主完成以下步骤。使用您的文件读取和写入工具检查会话数据并生成技能文件。

文件访问边界： 您所有的文件操作 **必须** 限制在此工作区目录内。工作区包含了您所需的所有数据——会话和技能已从共享存储复制到此处。请 **不要** 读取或写入工作区外的文件。服务器将从工作区收集您的修改并上传回存储。

步骤 1：阅读并理解会话数据

sessions/ 中的每个 JSON 文件都是一个 **预处理过的** 智能体会话。原始的交互日志已通过摘要

流水线压缩为紧凑格式。每个文件包含：

- `session_id`: 唯一标识符
- `task_id`: 本次会话尝试的基准任务
- `num_turns`: 原始会话的交互轮数
- `aggregate` (可选): 轨迹级统计量
 - `mean_score`: 多次滚动中的平均 ORM 得分
 - `success_count` / `fail_count`: 有多少次滚动尝试成功 / 失败
 - 稳定性: "all_success", "all_fail", 或 "unstable"
- `_skills_referenced`: 智能体读取或被注入的技能名称列表
- `_avg_prm`: 所有回合的平均 PRM 得分 (0.0–1.0; 越高越好)
- `_has_tool_errors`: 会话期间是否有任何工具调用失败
- `_trajectory`: **智能体动作的结构化逐步迹**。每一步显示: 使用的技能、工具调用及其参数和结果 (成功/错误)、智能体响应片段, 以及 PRM/ORM 得分。对于多轮对话, 每个轮次单独显示, 各自带有得分和成功标志。字段值被截断至 ~ 400 字符以保持紧凑——这已足以理解每一步发生的情况。
- `_summary`: **大模型生成的分析摘要** (8–15 句), 涵盖智能体的目标、策略、关键转折点、工具使用模式、技能有效性以及结果评估。

如何高效地阅读会话：

1. 以 `_summary` 开头, 快速概览每个会话。
2. 使用 `_trajectory` 以获取逐步的详细信息 (例如, 确定确切的工具调用失败原因, 或查看某个技能是如何使用的)。
3. 使用 `aggregate` 和 `_avg_prm` 进行跨会话的定量比较。
4. 使用 `_skills_referenced` 按技能对会话进行分组以完成步骤 2。

构建一个心理模型：

- 该智能体试图完成什么任务？
- 智能体成功了还是失败了？为什么？
- 提到了哪些技能？它们有帮助吗？
- 不同会话之间是否存在普遍的模式？

步骤 2：分析与汇总

按它们所提及的技能对小组会议进行分组：

- **技能组**: 参考特定技能的会话 → 评估该技能是否需要改进。
- **无技能会话**: 指未引用任何技能的会话 → 考虑是否应创建新技能。

对于每一组, 识别：

- 失效模式 (低 PRM 得分、工具错误、错误的方法)
- 成功模式 (高 PRM 得分, 有效使用工具)
- 故障是由**技能** (错误或缺失的指导)、**智能体** (误用、上下文上溢) 还是**环境** (API 不稳定、网络问题) 引起的。

第三步：阅读历史, 然后决定动作

在对现有技能做出任何动作之前，如果存在 `skills/<skill-name>/history/`，请阅读其下 **所有** 文件——每个 `v*.md` 和 `v*_evidence.md`。这是强制性的，而非可选的。你需要理解：

- 该技能在之前回合中的表现样子
- 为何之前进行了这些更改
- 哪些会议证据推动了这些变化？
- 先前的修改是否改善或降低了性能

只有在阅读完整的历史记录后，才能决定采取什么行动。否则，你可能会倒退先前的改进成果，或重蹈过去的错误覆辙。

在阅读历史时，请明确回答：

- 每个先验版本发生了什么变化？
- 什么证据支持了这一改变？
- 后续的会议是否表明这种改变起到了帮助、造成了损害，还是仍不明确？
- 在下一个版本中，哪些内容应保留，哪些应修订？

对于每个技能组，选择 **ONE** 个动作：

improve_skill 该技能内容需根据会话证据进行有针对性的修改。使用场景：

- 会议揭示了缺失的指导、过时的信息或模糊的指示
- 多个会话指向同一部分内容错误或不完整

optimize_description 技能主体尚可，但其描述导致了错误匹配。使用条件：

- 该技能正在被触发在不适用的任务上
- 仅需重写描述，无需修改正文

create_skill 会话证据揭示了一种反复出现的模式，该模式不属于任何现有技能。使用条件：

- 一种清晰且可传授的模式存在，该模式能够压缩与环境相关的知识
- 该模式足够独特，值得单独设立一项技能
- 现有技能无法涵盖此领域

skip 无需操作。用于：

- 这个技能效果还不错
- 证据太弱或模糊
- 失败是由智能体问题引起的，而非技能差距

有疑问时，优先跳过推测性修改。

第四步：执行——编写技能文件

对于 improve_skill / optimize_description：直接编辑现有的 `skills/<name>/SKILL.md` 文件。

对于 create_skill：创建一个新的目录 `skills/<new-name>/SKILL.md`。

SKILL.md 格式

每个 SKILL.md 文件都必须包含 YAML 前置元数据和一个 Markdown 正文：

```
---
name: lowercase-hyphenated-slug
description: What this skill does and when to trigger it. Include "NOT for: ..." exclusion
↳ conditions. 2-4 sentences.
category: general
```

<Markdown body with practical guidance>

步骤 5：保持技能历史记录

历史是演化记录簿——它记录了发生了什么变化、原因是什么，以及支持每个决策的证据。**每一个动作（创建、改进、优化描述）必须留下历史痕迹。**

重要：撰写前请先阅读

在接触任何现有技能之前，您必须：

1. 检查 `skills/<skill-name>/history/` 是否存在；如果存在，请列出它以查看所有现有的条目。
2. 如果存在，请阅读该目录中的每一个 **every** `v*.md` 和 `v*_evidence.md` 文件。
3. 如果存在，请了解完整的演变轨迹后再决定您的编辑。

跳过此步骤会导致严重错误——这将导致倒退之前的改进或与早期基于证据的决策相矛盾。

历史目录结构

```
skills/<skill-name>/history/  
  v0_evidence.md ← why this skill was created (for create_skill)  
  v1.md           ← SKILL.md snapshot before round 1 edit  
  v1_evidence.md ← sessions/feedback that drove the v1→v2 change  
  v2.md           ← SKILL.md snapshot before round 2 edit  
  v2_evidence.md  
  ...
```

历史命名规则

- 仅使用基于版本的文件名：`v<N>.md` 和 `v<N>_evidence.md`。
- **不要**使用日期、时间戳或随意命名的文件名，例如 `2026-04-04.md`、`notes.md` 或 `new_version.md`。
- 版本号必须反映技能的演化顺序，而非实际时间日期。
- 如果某个现有技能尚无历史记录，则您保存的第一个快照为 `v1.md`，对应的证据文件为 `v1_evidence.md`。

原因：实验每天可能进行多轮，基于日期的历史记录过于粗糙，无法准确还原每次编辑发生在哪个演化步骤中。

如何维护历史

为了提升技能 / 优化描述：

1. 检查 `skills/<skill-name>/history/` 以确定当前回合 N。如果没有历史记录，则为第 1 回合。
2. 将当前的 `SKILL.md` 内容原样复制到 `history/v<N>.md`。
3. 写入 `history/v<N>_evidence.md`，记录：
 - 哪些会话引发了这一变化（会话 ID、任务 ID、PRM 得分、成功/失败次数、工具错误、重复失败模式）
 - 正负信号是什么
 - 我查阅了哪些之前的记录以及这些记录如何影响了此次编辑

- 旧版本在已有会话证据中的表现如何
- 您正在保留、删除或修改哪些确切的条款或规则
- 你决定采取什么动作（改进 / 优化_描述）

4. 然后编辑 SKILL.md。

您的证据文件应呈现为简洁的版本化变更日志与性能评估，而非随意的笔记。确保未来智能体能够轻松回答：

- 为什么版本 v<N> 需要更改？
- 当前会话中的哪些证据支持下一步编辑？
- 先验版本在历史会话中表现如何？
- 哪些修改是故意进行的，不应随意撤销？

对于 create_skill：不存在之前的版本，但仍需编写 history/v0_evidence.md，说明：

- 哪些会话激发了创建（ID、得分、失败模式）
- 为什么现有的技能无法覆盖这种模式
- 你决定采取什么动作（create_skill）

证据文件内容期望

每个 v<N>_evidence.md 文件应以简洁但明确的形式包含：

1. 决策摘要

- 动作类型
- 目标技能
- 现在为何需要改变

2. 会话证据

- 相关会话 ID / 任务 ID
- 代表性 PRM 得分或综合指标
- 反复出现的工具故障/观察结果

3. 历史比较

- 此前版本尝试过
- 即使后续证据表明这些修改改善了结果、导致结果恶化，或仍无法得出明确结论

4. 编辑计划

- 技能被更改的具体部分
- 确切的部件被有意保留

5. 开放性问题

- 未来回合应予以关注的不确定性

历史持久性取决于新式模式

- 在 --no-fresh 模式下，服务器每轮都会从存储中刷新 SKILL.md，但不会清除 history/ 子目录。因此，历史记录会在各轮之间持续累积，形成连续的审计追踪。
- 在 --fresh 模式下，每次迭代都会从头开始重建工作区，因此本地的 history/ 目录不会自动保留。除非当前工作区已包含历史文件，否则请将每次迭代视为独立的演化过程。

编辑原则

保守编辑（用于 improve_skill）

- 将当前技能视为**真理来源**，而不是草稿。
- 默认采用**目标编辑**，而非重写。
- 保持原始结构、标题顺序和术语。
- 如果失败情况仅为边缘案例，请添加缺失的检查或明确约束，而无需更改无关部分。
- 只有当证据表明其内容存在实质性错误时，才重新编写整个部分。
- 如果成功会话支持某个部分，则除非失败证据明确与之矛盾，否则保持不变。

硬约束

- 请勿更改 API 合同、端口、端点、输出路径、有效负载格式或所需文件名——除非会话证据明确显示它们已发生变化。
- 请勿移除与观察到的故障无关的核心功能、API 参考或工具使用示例。
- 请勿将一项技能转换为具有不同用途的另一项技能。
- 不要从头开始重写整个技能。
- 除非证据要求，否则不要强加新的模板或写作风格。
- 除非环境有特殊之处，否则请勿添加通用的最佳实践建议（重试逻辑、缓存、状态管理）。

技能问题与智能体问题的区分

并非每一次失败都是技能不足造成的：

- **技能问题**（错误/缺失指引）→ 编辑技能。
- **智能体问题**（误用、重启、上下文上溢）→ 请勿在技能中加入智能体运行时的建议。
- **环境问题**（API 不稳定性，网络不稳定性）→ 若重复出现，请添加简要说明，但保持简洁。

关键反模式：如果技能 **ALREADY** 已包含正确的环境信息，而智能体因未使用该信息而导致失败，这属于 **AGENT** 问题。请勿删除正确的 API 信息，也勿用“去检查源代码”之类的指令进行替换。

技能编写原则（用于 `create_skill`）

- 一个技能应该压缩 **环境信息**（API 端点、端口、负载格式、工具特性、领域流程）——而不是智能体已经知道的通用最佳实践。
- 偏好简短、以动作为导向的名称（小写连字符格式）。
- 名称 **必须** 与所有现有技能不同。创建新技能前，请检查 `manifest.json` 中当前的技能名称列表。
- 描述是主要的触发机制——请在其中明确写出触发条件，包括“NOT for: ...”的排除条件。
- 内容应具有领域针对性、实用价值且不显而易见。
- 请使用命令式指令。自然地组织正文以完成任务。
- 在任务核心部分包含具体的 API 端点、端口、命令模式和数据负载示例。
- 保持简洁、可复用且基于证据。
- 编写可复用的指导原则，而非失败总结或事后分析。

注意事项

- 如果证据支持，您可以在一次会话中创建多个技能。
- 处理所有会话——在第一组之后不要停止。
- 直接将您的更改写入 `skills/` 目录中的文件。服务器将通过比较文件哈希值来检测变更。

- **始终**在决定对该技能采取任何动作之前，阅读 `skills/<name>/history/` 中的 **所有** 文件（如果该历史目录存在）。这是强制性的，而非可选的。
- **始终**在更改之前保存旧版本和证据。
- **始终**使用基于版本的历史文件名 (`v<N>.md`, `v<N>_evidence.md`)；永远不要使用基于日期的文件名。
- 请勿修改 `sessions/` 中的文件——它们是只读输入。
- 请勿修改 `manifest.json` 或 `skill_registry.json` — 由服务器管理这些文件。
- 请勿访问此工作区目录之外的文件。
- 如果会话中没有可操作的模式，完全不必做任何更改。