VISE：与验证器无关的技能演化：来自工具实现的反馈

YuxuanLiu[1] ChongyangDu[2] Yuhao Zhang[2] JiaheGuo[3]Kawai Chung[1] Wenbin Hu[1] Huihao Jing[1] Hongyu Luo[1]Qing Zong[1] Haochen Shi[1] Haoran Li[1†] Yangqiu Song[1]

[1]香港科技大学，中国香港特别行政区，中国[2]哈尔滨工业大学，深圳，中国[3]哈尔滨工业大学，哈尔滨，中国 {yliurk,hlibt}@connect.ust.hk, yqsong@cse.ust.hk

[†]通讯作者。

arXiv:2611.32480v1 [cs.AI] 24 Nov 2026

摘要

智能体技能提供可复用的工作流，引导大语言模型智能体使用工具。改进这些工作流需要针对缺失或错误指令的反馈，但特定于任务的验证器并不总是可用。我们提出 VISE（与验证器无关的技能演化），这是一种脚本优先方法，利用工具实现来指导技能修订。该方法从源代码中提取工具的能力、输入要求、使用条件和输出格式。借助任务请求和执行轨迹，它将这些信息与当前工作流指令进行比较，以识别缺失或冲突的步骤。随后，它为 SKILL.md 推荐具体编辑，并将每项修改关联到相关实现和使用条件。我们在 SkillsBench 和 SkillLearn-Bench 上，将 VISE 作为四种演化方法的额外修订阶段进行评估。与相应的无验证器条件相比，它使总体平均任务成功率提高了 9.14–13.35 个百分点。三次运行中全部解决的任务比例提高了 12.36–18.29 个百分点。一项案例研究展示了修订后的指令如何将工具输出连接到后续步骤。这些结果表明，工具实现可以作为技能自我进化的实用反馈来源。

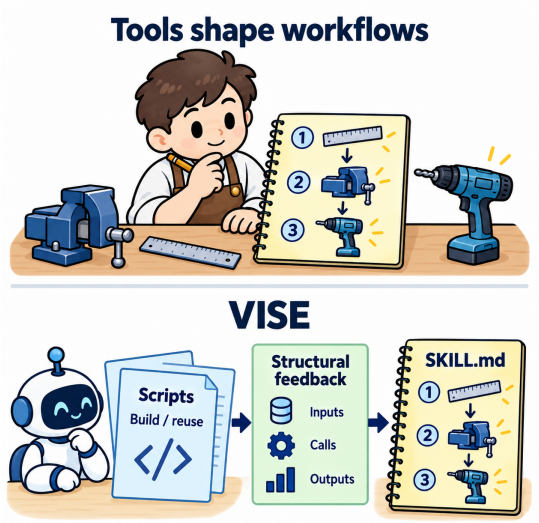

图1：VISE 背后的脚本优先理念。人们围绕工具组织工作流。VISE 同样会构建或复用脚本，并从其实现中提取结构化反馈，
以完善 SKILL.md。

1 引言

智能体技能提供可复用的程序，帮助大语言模型智能体使用工具完成任务（Zhang 等人，2025；Li 等人，2026年）。有效的技能会明确说明要使用哪些工具、如何准备其输入，以及如何利用其输出。例如，目标计数工作流必须说明要处理哪些视频帧，以及如何汇总返回的计数结果。缺少这些指导时，智能体可能会调用工具，却不利用其输出完成任务。技能自我进化通过执行经验不断优化这类指令和反馈（Zhang 等人，2026a；Liu 等人，2026c）。因此，反馈必须明确指出要修改哪些指令以及如何修改。

现有方法借鉴了模型生成的反馈、执行经验和任务评估。Self-Refine 使用自反馈来改进输出（Madaan 等人，2023年），而 SkillLearn-Bench 研究了针对技能的自反馈（Zhong 等人，2026年）。Reflexion 将执行经验转化为文字反思，以指导后续尝试（Shinn 等人，2023年）。基于验证器的方法使用成功判定或诊断报告来指导技能更新（Zhang 等人，2026a）。SkillRevise 还会重新执行修订后的技能，以从观测到的候选项中选出最佳者（Liu 等人，2026c）。相关工作从代码中提取程序性知识：Repo-To-Skill 将代码仓库知识提炼为可复用的技能（Chen 等人，2026年）。

当任务验证器反馈不可用时，智能体必须根据任务要求、工具实现和执行历史确定如何修订技能。然而，可用工具可能仅支持任务的一部分，或者要求现有指令未涵盖的输入和条件。因此，有效的修订需要确定哪些工具能力适用于该任务，以及这些工具的要求应如何影响工作流。随后，智能体必须识别现有指令中遗漏必要步骤或与实现相冲突之处，并将这些缺口转化为可执行的修订方案。

我们提出了 VISE（与验证器无关的技能演化），这是一种脚本优先方法，利用工具实现来指导技能修订。对于新技能，它首先选择或创建与任务相关的脚本，然后在 `SKILL.md` 中编写初始指令。修订时，它会提取有关工具能力、使用条件、步骤和约束的信息，并引用相关源代码作为依据。它会结合任务请求和执行轨迹，将这些信息与当前指令进行比较。对于每个缺失或冲突的步骤，它都会提出具体的修改建议，并说明适用条件。修订阶段会更新 `SKILL.md` 中的相关内容，同时保留脚本。此阶段也可用于改进其他演化方法生成的技能。图 1 展示了这一方法。

我们在 SkillsBench 和 SkillLearnBench（Li 等人，2026年；Zhong 等人，2026年）上评估 VISE，进一步修订 Self-Refine、TraceEvo、SkillLearn 和 SkillRevise 的无验证器输出。与无验证器条件相比，这四种方法的总体任务平均成功率提高了 9.14–13.35%。三次运行中均能解决的任务比例提高了 12.36–18.29%（表 2）。在总体指标上，VISE 的表现也优于二元、报告和诊断条件。视频计数案例研究展示了修订后的指令如何将工具输出与后续步骤联系起来。在另一项研究中，重复评估表明，相似的总体收益可能掩盖受益任务的变化。新构建的 SkillsBench 变体进一步表明，要求的变化可能会削弱甚至逆转在原始任务上观察到的技能收益。

我们的贡献主要有三点：
1. 工具 grounding 的技能演化。我们提出了 VISE，将工具实现中的信息与缺失或相互矛盾的工作流指令关联起来，并生成有针对性的修订反馈。**2. 对反馈来源进行系统比较**。我们在 SkillsBench 和 SkillLearnBench 上，对四种技能演化方法的反馈来源进行比较，衡量平均任务成功率、任务覆盖率和重复执行成功率。

3. 技能效用的实证研究。通过反复评估和任务变体，我们分析哪些任务能够从使用技能中受益，以及这些收益如何随任务要求的变化而改变。

2 相关工作

技能自演化。SkillLearnBench（Zhong 等人，2026年）研究智能体如何通过自反馈在没有外部监督的情况下修订技能。SkillEvolver（Zhang 等人，2026a）在部署失败后修订指令和代码，而 SPARK（Zhou 等人，2026年）从环境验证轨迹中提炼技能。SkillRevise（Liu 等人，2026c）将缺陷诊断与检索到的修复原则相结合，并重新执行修订后的技能，以选出观测到的任务效用最高的候选项。EvoSkill（Alzubi 等人，2026年）中的任务评估同样用于指导候选项选择；SkillOpt（Yang 等人，2026年）则利用任务评估控制有界的技能编辑。CoEvoSkills（Zhang 等人，2026b）在演化过程中结合替代验证与预言机反馈。多轮分析表明，技能改进十分稀疏，并且取决于修订过程中所展示的反馈（Liu 等人，2026d）。这些方法侧重于源自执行的反馈，而不是利用工具实现来修订现有的工作流指导。

工具支撑的程序性知识。LATM（Cai 等人，2024年）和 CREATOR（Qian 等人，2023年）将工具创建与工具使用分开。Voyager（Wang 等人，2023年）构建可执行技能库，而 AutoManual（Chen 等人，2024年）则将交互衍生规则整理为规划手册。DRAFT（Qu 等人，2025年）和 Play2Prompt（Fang 等人，2025年）通过探索性交互改进工具文档，其中 Play2Prompt 生成工具使用示例以供验证。DocsChisel（Lu 等人，2026年）利用失败执行轨迹修订文档字段，并依据验证性能选择修订版本。Repo-To-Skill（Chen 等人，2026年）将代码仓库知识提炼为技能，明确能力、使用条件和程序。SkillScriptBench（Liu 等人，2026a）分别评估文档修复、脚本修复和保留能力。总体而言，这些研究通过获取、文档改进、包维护和工作流评估来考察程序性知识。

方法	原始版本	$\lvert\Delta\rvert$	变体
无技能	41.70	8.08	41.09
Skill-Creator	37.70	8.91	41.76
SkillEvolver	62.78	0.87	53.81
SPARK	38.78	4.30	42.26
CoEvoSkills	49.44	6.00	36.60
Self-Refine	60.46	0.16	58.55
SkillLearn	55.11	1.36	57.80
Trace Evo	61.82	3.87	58.84
SkillRevise	57.57	0.28	54.36

表1：SkillsBench及其变体的平均奖励（0–100）。原始版本取两次评估的平均值；$\lvert\Delta\rvert$为两者的绝对差值（附录A）。

3 技能效用的实证研究

在需要修订技能时，任务专用验证器可能无法使用，因此有必要研究替代性反馈来源。评估这些方法需要区分执行性能与使用技能所带来的收益。我们通过在 SkillsBench（Li 等人，2026年）上进行重复评估，以及使用修改输入表示、任务实例或耦合要求的任务变体，来考察这种区别。

3.1 性能提升的来源

执行得分较低并不一定意味着使用技能带来的收益较小。CoEvoSkills 的得分在两次评估中从 52.44 降至 46.44，但相较于配对的无技能参考基线，其增益仍然相近，分别为 10.24 和 9.96 分。类似的汇总结果也可能掩盖成功任务的变化。Self-Refine 在每次评估中都有 50 个任务获得完整奖励，但其中 9 个任务失去了完整奖励，另有 9 个任务获得了完整奖励。更多任务层面的比较见附录 A.1。

3.2 不断变化的任务需求下的效用

我们构建了三类 SkillsBench 变体，以检验技能在原始任务之外是否仍然有效。表征变更（*P1*）会修改输入或输出表征，例如文件路径、字段名称或序列化格式。实例变更（*P2*）会替换具体的任务实例，例如数据记录或授权关系。耦合需求变更（*P3*）会改变彼此依赖、必须保持一致的约束或输出。每种方法的技能都在原始任务上生成或演进，并在不进一步修订的情况下于这些变体上进行评估。

如图所示 2，SkillEvolver 和 Co-EvoSkills 在三类任务中的得分均低于其原始任务平均分

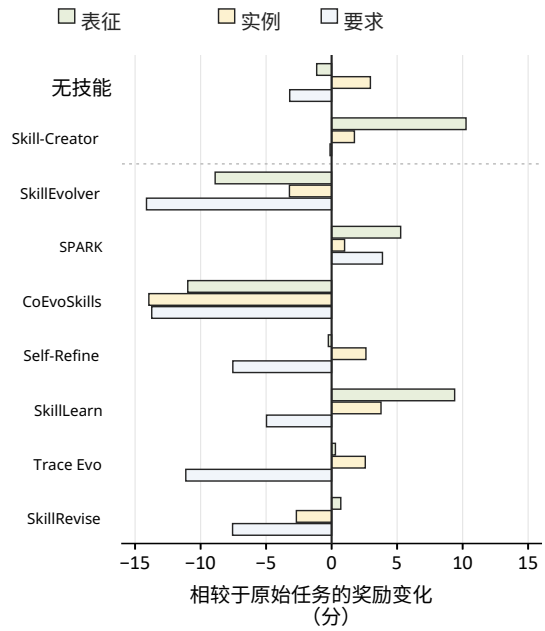

图 2：各任务变更类别下的奖励变化。每个条形图均以方法在表1中的原始版本均值减去其在相应变体类别中的均值。

，而 SPARK 的得分则高于原始任务平均分。对于 Self-Refine、SkillLearn、Trace Evo 和 SkillRevise，耦合要求变体的下降幅度最大。不过，如表 1 所示，这四种方法的总体变体得分仍高于 Skill-Creator 和无技能。

3.3 修订证据与研究问题

这些发现促使我们从总体表现和任务级效益两方面评估技能修订。此外，进行修订还需要确定工作流中应改变哪些内容：调用哪些操作、如何准备其输入，以及如何使用其输出。公开的工具实现展示了这些操作及其约束，为考察相应的工作流指导提供了依据。我们通过以下研究问题，探究这些实现知识能否在没有任务验证器反馈的情况下支持有效的技能修订。

研究问题 1. 不同演化方法和任务类别下，反馈来源有何差异?

研究问题2. 没有任务验证器反馈时，基于工具的反馈能否提升技能?

研究问题3。 实施证据如何体现在修订后的指导意见和工具使用中?

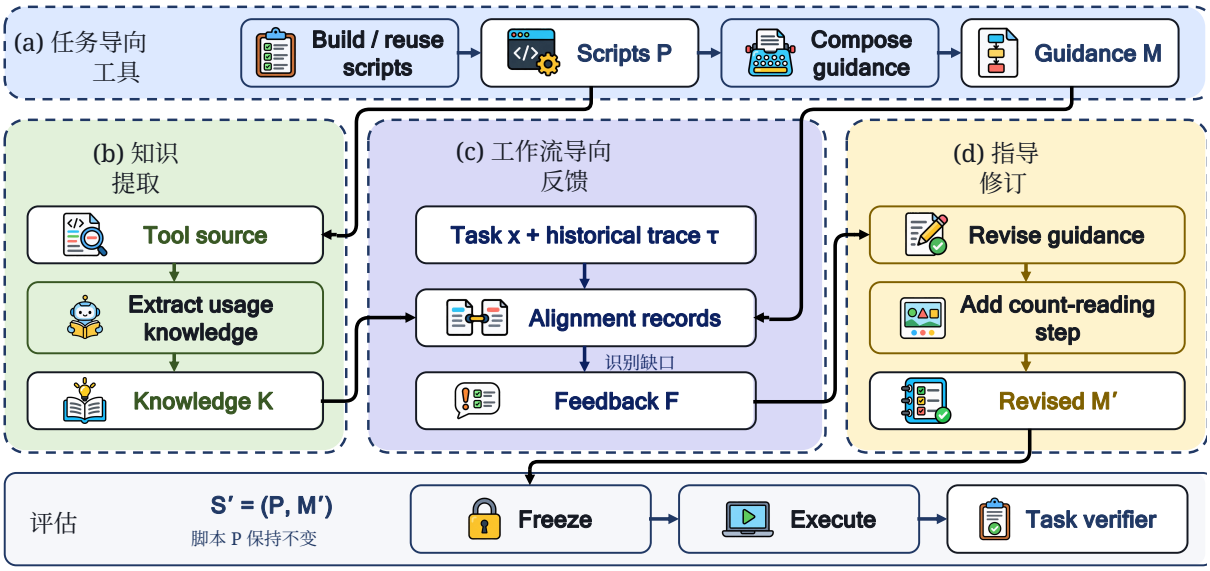

图3：VISE 使用可执行工具来构建和修订工作流指导。脚本 P 支持初始指导M_0；修订可以从 M_0 或演进后的版本 M开始。操作知识 $\mathcal{K}$ 与任务 x下的 M 及轨迹 τ 相对齐，以识别指导缺口并形成反馈 $\mathcal{F}$，从而进行有针对性的编辑，例如添加逐对计数处理。修订后的指导意见 M' 与保持不变的脚本 P 共同构成 S'，并在下游评估前冻结。

4 VISE：从工具演化工作流

VISE 遵循脚本优先范式，实现不依赖验证器的技能演进：可执行工具提供工作流构建所围绕的操作，而其实现则为修订工作流指导提供证据。我们将一项技能表示为 $S = (P, M)$，由可执行脚本 P 和 M 中的指导意见组成 `SKILL.md`。VISE 首先确立 P，并围绕其操作编写初始指导。随后，它从 P中提取操作知识，在任务情境下将这些知识与M 进行比较，并构建结构化反馈，以便有针对性地修订，而不依赖任务验证器判定。图 3 展示了这四个阶段。

4.1 构建任务依托工具

VISE 首先建立其工作流将围绕组织的可执行操作，从而构建初始技能。给定任务请求 x，它通过复用合适的实现或生成缺失的脚本，构建脚本集 P。这些脚本实现具体的任务操作，并公开其输入、可配置参数和输出格式，为编排工作流指导提供操作基础。

VISE 随后根据 M_0、x 和 P 生成初始指导。该指导说明如何为可用操作准备输入、安排调用顺序，并在后续步骤中使用中间输出。脚本负责实现这些操作，而指导则协调它们的使用，以处理任务请求。二者共同构成初始技能 $S_0 = (P, M_0)$。

4.2 提取任务依托工具的操作知识

初始指导 M_0 将可执行操作组织成任务工作流。为支持后续修订，VISE 从脚本 P 中提取操作知识，使其使用步骤、条件和约束明确化，以便与指导 M 进行比较。语言模型从完整源代码中推导这些知识；在支持解析的情况下，还会借助由定义和参数、调用表达式及条件分支构成的语法索引。

相关实现细节被组织成工作流级记录 $\mathcal{K} = \{k_1, \ldots, k_m\}$。每条记录将某项操作所提供的内容与使用该操作的条件和步骤联系起来：

$$k = (\text{cap}, \text{cond}, \text{steps}, \text{limits}, \text{src}), \quad (1)$$

其中，各字段分别表示能力、适用条件、有序使用步骤、实现约束和支持性源引用。这种表征将分散在实现中的信息整合为对相关操作使用方式的描述。

例如，对象计数记录结合了图像和模板输入、匹配与

去重参数，并通过使用打印输出获取计数。因此，它既描述了如何调用该操作，也描述了如何访问其结果。模型将每条记录与支持性源文件及行范围关联起来；VISE 会检查这些位置并附加相应的代码。

4.3 构建以工作流为依据的反馈

VISE 将操作知识 $\mathcal{K}$ 与指导 M进行比较，并以任务请求 x 和历史执行轨迹 τ 作为上下文。对于与任务相关的工作流，模型会检查 M 是否规定了必要的操作、是否反映了这些操作的使用条件，以及在适用实现限制时是否提供了替代步骤。

模型将比较结果整理为对齐记录，连接任务要求、现有指令、支持性知识和已识别的指导缺口。需要修改的记录会生成反馈 $\mathcal{F} = \{f_1, \ldots, f_n\}$。每个条目都会描述问题、建议修改方案，并说明适用条件，同时引用支持性知识和任务要求。对于纠正内容，还会引用 M中存在冲突的段落。

在对象计数示例中， M 提供了检查脚本并使用运行时证据的一般指令，而 $\mathcal{K}$ 描述了工具的输入和计数输出。由此产生的反馈针对缺失的调用和输出处理步骤，建议明确指定调用参数，并读取每个帧–模板对所打印的计数。

4.4 修订工作流指导

给定任务请求 x、操作知识 $\mathcal{K}$ 和反馈 $\mathcal{F}$，修订模型会对指导 M 生成有针对性的编辑，同时保留现有的有用程序。每项编辑都将原始段落与替换指令配对，并注明其所针对的反馈项。

VISE 会检查原始段落是否与 M 中的唯一位置相匹配，以及各项编辑是否互不重叠。随后，它应用替换内容以生成 M'，并保持脚本 P 不变：

$$S = (P, M) \longrightarrow S' = (P, M'). \qquad (2)$$

修订后的技能 S' 在下游执行和任务验证器评估之前即被冻结。编辑限制和验证细节见附录 B。

5 实验

我们在 SkillsBench 和 SkillLearn-Bench 上评估 VISE，以考察工具支撑的反馈能否在修订过程中不使用任务验证器判断的情况下改进技能。在四种演化方法中，我们将经 VISE 精炼的技能与相应的无验证器技能以及借助任务验证器反馈演化的技能进行比较。我们度量平均成功率、任务覆盖率和重复成功，以刻画其下游效用。

5.1 实验设置

任务与初始技能。 我们在 Skills-Bench 和 SkillLearnBench（Li 等人，2026年；Zhong 等人，2026年）上进行评估。对于每项任务，我们从一项初始技能 $S_0 = (P, M_0)$ 开始技能演化，其中包含公开的可执行脚本 P 和初始工作流指南 M_0。我们将 Skill-Creator（Anthropic，2026年）作为未演化基线。

演化方法。 我们评估了四种技能自演化方法。Self-Refine 基线（Madaan 等人，2023年）直接修订工作流指导 M ，使用`SKILL.md` 中的可用输入。TraceEvo 通过分析执行轨迹来找出指导中的不足。SkillLearn（Zhong 等人，2026年）检查任务请求和现有技能，以找出缺失或不准确的指令。SkillRevise（Liu 等人，2026c）诊断技能缺陷，并检索修复原则以指导修订。对于 SkillLearn 和 SkillRevise，我们还评估了它们各自至多进行一次修订的原生工作流，在表2中标记为 Native。技能演化和下游执行均使用 GPT-5.5。

反馈条件。 所有条件均使用任务请求 x、脚本 P、工作流指导 M 和历史执行轨迹 τ。无验证器条件仅使用这些输入。二元条件会额外加入验证器对初始执行作出的通过/失败判定。报告条件还会提供测试失败报告，其中包括可用的预期值和观测值。诊断条件会提供判定以及原生评估元数据、诊断消息和失败轨迹。VISE 会提供结构化反馈$\mathcal{F}$，将从 P 中提取的操作知识 $\mathcal{K}$ 与指导 M 对齐，并以 x 和 τ 作为上下文。指导修订期间脚本保持不变。

评估。 对于每项任务，我们冻结最终技能，并使用基准测试的原生验证器对其进行三次下游执行评估。遵循任务覆盖率与

方法	条件	SkillsBench			SkillLearnBench			总体		
		P@1	P@3	Hit^3	P@1	P@3	Hit^3	P@1	P@3	Hit^3
Skill-Creator	—	39.68	53.57	25.00	39.39	54.55	24.24	39.53	54.10	24.59
Self-Refine	无验证器	37.30	51.19	22.62	40.74	58.59	25.25	39.16	55.19	24.04
	二元	38.49	53.57	22.62	36.36	50.51	21.21	37.34	51.91	21.86
	报告	40.08	53.57	26.19	46.13	**63.64**	27.27	43.35	59.02	26.78
	诊断	45.63	**63.10**	28.57	47.81	61.62	33.33	46.81	**62.30**	31.15
	VISE	**50.81**	57.32	**42.68**	**49.14**	61.86	**36.08**	**49.91**	59.78	**39.11**
TraceEvo	无验证器	38.10	54.76	22.62	40.07	54.55	25.25	39.16	54.64	24.04
	二元	37.30	54.76	21.43	43.10	61.62	23.23	40.44	58.47	22.40
	报告	39.29	50.00	27.38	43.43	61.62	27.27	41.53	56.28	27.32
	诊断	40.87	**60.71**	23.81	45.45	63.64	30.30	43.35	62.30	27.32
	VISE	**52.03**	58.54	**43.90**	**52.92**	**65.98**	**37.11**	**52.51**	**62.57**	**40.22**
SkillLearn	无验证器	38.10	53.57	22.62	39.06	55.56	23.23	38.62	54.64	22.95
	二元	36.90	47.62	23.81	43.43	60.61	24.24	40.44	54.64	24.04
	报告	40.48	54.76	23.81	45.79	**65.66**	24.24	43.35	60.66	24.04
	诊断	38.49	55.95	21.43	39.73	52.53	28.28	39.16	54.10	25.14
	VISE	**51.44**	**58.02**	**44.44**	**50.69**	64.58	**38.54**	**51.04**	**61.58**	**41.24**
	原生	51.41	55.42	45.78	52.53	56.57	46.46	52.01	56.04	46.15
SkillRevise	无验证器	43.25	59.52	27.38	41.41	56.57	29.29	42.26	57.92	28.42
	二元	43.25	53.57	33.33	42.42	59.60	28.28	42.81	56.83	30.60
	报告	47.62	**60.71**	33.33	**53.20**	**68.69**	37.37	50.64	**65.03**	35.52
	诊断	50.79	**60.71**	39.29	48.82	63.64	35.35	49.73	62.30	37.16
	VISE	**51.22**	58.54	**42.68**	51.55	67.01	**39.18**	**51.40**	63.13	**40.78**
	原生	50.20	56.63	42.17	57.91	64.65	47.47	54.40	60.99	45.05

表2：下游成功率（%）。粗体表示每种方法内五种反馈条件中按列取得的最佳分数；灰色行表示 VISE。原生报告展示了每种方法的原始工作流，以供比较。总体结果对各任务赋予相同权重。有关任务覆盖率和评估协议，请参见附录 C 。

重复可靠性（Yao 等人，2024）之间的区分，我们将 P@1 报告为平均成功率，将 P@3 报告为至少成功一次的任务比例，并将 Hit^3报告为三次执行均成功的任务比例。结果按每个基准测试分别报告，并以每项任务等权重汇总为总体结果。评估方案和任务覆盖率详见附录C。

5.2 主要结果

不同演化方法的反馈效果各不相同。如表2所示，在无验证器条件下，只有 SkillRevise 的总体 P@1 高于 Skill-Creator。与无验证器相比，二元反馈会降低 Self-Refine 的总体 P@1，而报告反馈会使四种方法的总体 P@1 提高 2.37–8.38%。将报告反馈切换为诊断反馈，会提高 Self-Refine 和 TraceEvo 的总体 P@1，但会降低 SkillLearn 和 SkillRevise 的总体 P@1。与无验证器相比，在四种演化方法中，VISE 的总体 P@1、P@3 和 命中率^{3}均更高，增幅分别为 9.14–13.35%、4.59–7.93% 和 12.36–18.29%。

VISE 的总体 P@1 和Hit^3均高于三种任务验证器反馈条件。如表2所示，VISE 在每项指标上均超过最佳任务验证器条件：P@1 提高 0.76–9.16%，Hit^3, 提高 3.62–16.10%，且在每种演化方法中，Hit^3 的优势幅度更大。总体 P@3 的排名因方法而异，其中 VISE 在 TraceEvo 和 SkillLearn 中取得最高分。对于 SkillLearn 和 SkillRevise 的原生工作流，比较结果有所不同：原生工作流的总体 P@1 和Hit^3, 更高，而 VISE 的总体 P@3 更高。

6 分析

6.1 研究问题 1：不同方法和任务类别中的反馈效果

VISE 和任务验证器反馈的相对效果因任务类别而异。如图4所示，在 SkillsBench 的数学与运筹学类别以及 SkillLearnBench 的实用工具与其他类别中，四种演化方法下，VISE 的 P@1 均高于三种任务验证器反馈条件。在 SkillLearnBench 的软件工程类别中，VISE 与 Report 的优劣顺序则相反：Report 相较于无验证器的提升为 5.9–21.6%，而 VISE 带来的变化为 -3.9%至 3.9%。

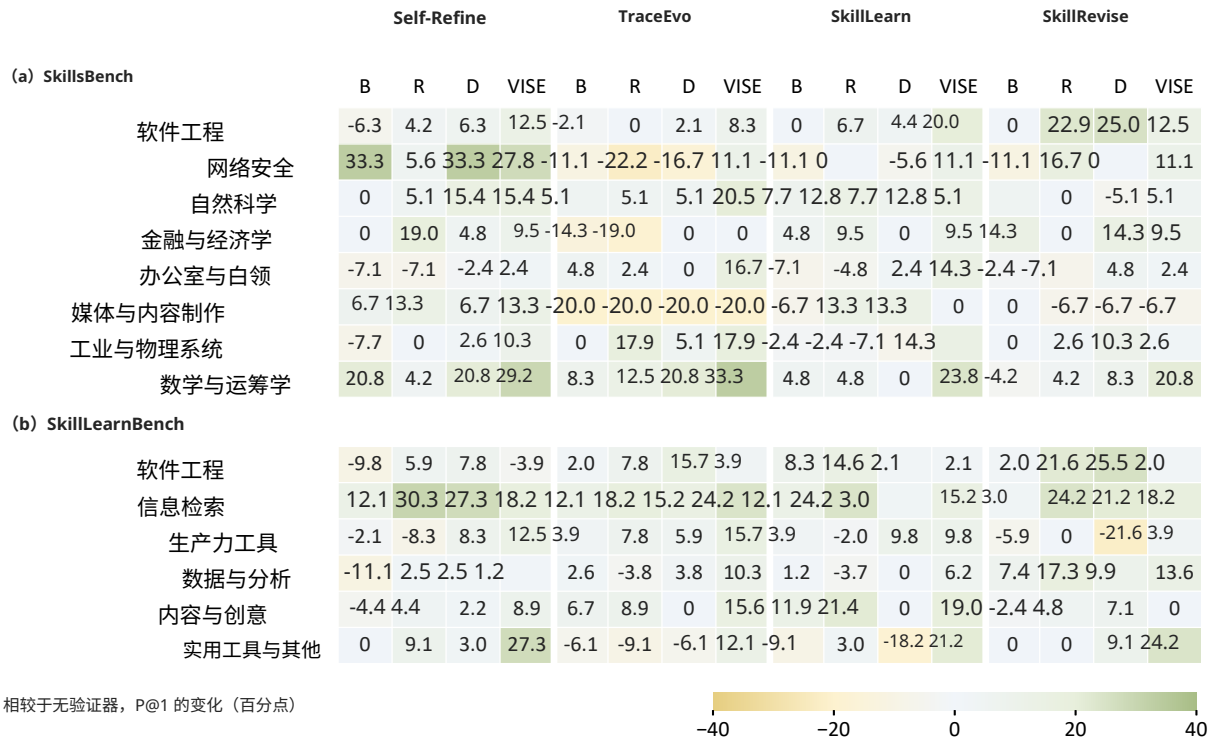

图4：反馈增益因任务类别和演化方法而异。单元格显示表 2’中各方法共同完整任务集上无验证器条件下的 P@1 变化（百分点）。B/R/D：二元/报告/诊断。

在同一任务类别中，反馈带来的收益也取决于演化方法。在 SkillLearnBench 的生产力工具类别中，对于 Self-Refine、TraceEvo 和 SkillLearn，诊断的 P@1 高于无验证器；但对于 SkillRevise，P@1 则降低了 21.6%。相比之下，在该类别中，四种方法下 VISE 的 P@1 均高于无验证器，提升幅度为 3.9–15.7%。

6.2 研究问题2：无任务验证器反馈的技能改进

VISE 将成功分布转向三次执行均解决的任务。如图6所示，在 VISE 下，仅有一次或两次执行成功的任务占 20.3%–22.3%，而在无验证器设置下这一比例为 29.1%–32.2%。在共同的SkillLearn任务集上，VISE 和报告的 P@3 均为 61.58%，但在 VISE 下，至少成功一次的任务中有 67.0% 在三次执行中均成功，而报告下这一比例为 40.4%。因此，相同的任务覆盖率掩盖了不同的重复成功模式。

修订后的技能在迁移到未经进一步修订的任务变体时，也仍然具有优势。我们冻结在原始任务上修订的技能，并直接在第3.2节中的 233 个 Skills-Bench 变体上对其进行评估。如

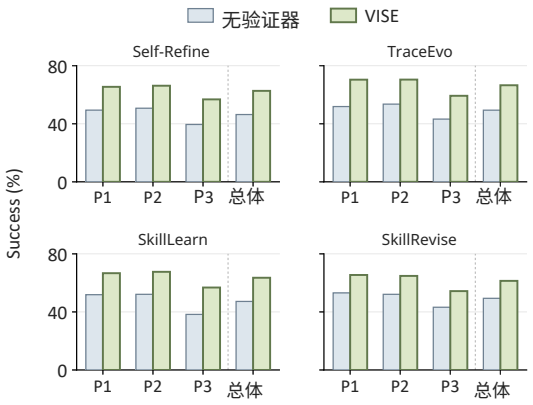

图5：VISE 提升了向任务变体的迁移能力。P1/P2/P3：表征、实例和耦合要求的变化。

图5显示，VISE 在四种方法中均提升了三个变体类别的成功率。总体成功率从 46.35%–49.36% 上升至 61.37%–66.52%。即使经过修订，耦合要求变体仍是每种方法得分最低的类别。

6.3 研究问题3：指导和工具使用中的实施证据

我们重新审视第 4 节中的 SkillLearnBench 视频物体计数示例，以考察

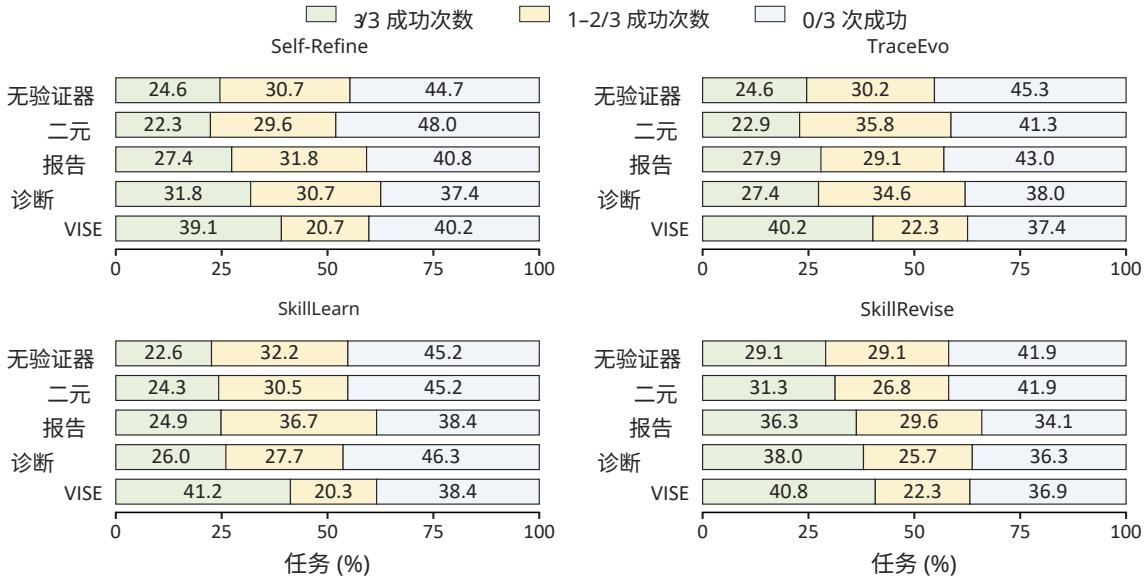

图6：不同反馈条件下的成功次数分布。条形图显示了在三次评估记录中，成功三次、一次或两次以及零次的任务比例。五种条件在每种方法中使用相同的任务。

修订后的指导意见如何体现在执行过程中。VISE 将 Self-Refine 技能中检查脚本的一般指令扩展为计数过程，明确说明如何调用工具，以及如何使用每个帧–模板对对应的打印输出。

两次执行都会调用计数器，但将其以不同方式整合到工作流中。在原始版本的执行过程中，记录的调用会在 CSV文件创建完成后，处理单个帧。使用修订后的技能时，智能体会在帧–模板循环中调用计数器，解析其打印的计数结果，并将其汇总到最终 CSV中。此次执行遵循新增的输出使用指令：计数器的结果成为最终结果构建的输入。

6.4 组件比较

条件	P@1	P@3	Hit^3
无工作流修订	↓8.33	0.00	↓5.00
不含结构化反馈	↓13.89	↓16.67	↓5.00
不含工作流对齐	↓10.56	↓5.00	↓1.67
原始来源修订	↓18.89	↓13.33	↓16.67
完整 VISE	57.78	76.67	30.00

表3：TraceEvo 在60个任务上的组件比较。完整 VISE 报告绝对分数（%）；其他行报告相对于完整 VISE 的百分点下降。

我们从原始基准测试中选取了 60 个附带脚本的任务。所有组件比较均使用 TraceEvo。完整的 VISE 获得了更高的 P@1，且命中率^3 高于无工作流修订的情况，而 P@3 保持不变，如表3所示。在至少成功解决过一次的任务中，平均成功次数从 1.93 增至 2.26：P@1 的提升反映了重复成功次数增加，而非任务覆盖率扩大。完整的 VISE 还至少成功解决了 46 个任务，而直接修订源码和轨迹且不提供结构化反馈的做法仅解决了 36 个任务。

与无工作流对齐相比，完整 VISE 在 P@1 上的优势大于在 Hit^3,上的优势，其中一次成功的任务更少、两次成功的任务更多，而三次成功的任务数量相近。条件定义和完整的成功次数分布见附录 C。

7 结论

VISE 将技能演化建立在可执行工具之上：脚本确定可用的操作，而其实现则为修订工作流指导提供依据。我们的实证研究表明，相似的总体增益可能惠及不同的任务，而任务需求的变化可能削弱或逆转技能带来的收益。在 SkillsBench 和 SkillLearn- Bench 上采用四种演化方法时，与相应的无验证器条件相比，VISE 提高了平均成功率、任务覆盖率以及三次运行成功率。这些结果表明，与验证器无关的技能演化可以通过明确连接工具能够执行的操作与工作流指导如何引导其使用而获益。

局限性

VISE 从可检查的工具实现中获得修订反馈。对于依赖闭源工具或远程 API 的技能，可访问的脚本可能会暴露调用逻辑，但只能提供有关底层操作的有限信息。这限制了可用于工作流修订的操作知识。将工具文档和观察到的执行行为纳入其中，是将 VISE 扩展到这些场景的自然方向。

伦理考量

修订后的技能指导可能会在不同任务之间重复使用，从而使不安全指令或敏感信息传播到原始交互之外。在安全敏感的场景中部署时，应保留工具级访问控制，并在重复使用更新内容前进行审核。共享成果应保留来源归属信息，并排除凭据和私人任务数据。

参考文献

Salaheddin Alzubi、Noah Provenzano、Jaydon Bingham、Weiyuan Chen 和 Tu Vu。2026年。EvoSkill：多智能体系统的自动化技能发现。*arXiv* 预印本 *arXiv:2603.02766*。第1版。

Anthropic。2026年。Skill-Creator。软件文档。访问日期：2026年10月13日。

Tianle Cai、Xuezhi Wang、Tengyu Ma、Xinyun Chen 和 Denny Zhou。2024 年。大语言模型作为工具制造者。载于国际学习表征会议。

Jianlyu Chen、Yuyang Hu、Hongjin Qian、Jiawei Liu、Wenqing Wei、Xiaolong Chen、Defu Lian、Zhicheng Dou、Chaozhuo Li、Qiwei Ye 和 Zheng Liu。2026年。Repo-To-Skill：将 GitHub 仓库提炼为 AI4AI 技能。*arXiv* 预印本 *arXiv:2609.02749*。第1版。

Minghao Chen、Yihang Li、Yanting Yang、Shiyu Yu、Binbin Lin 和 Xiaofei He。2024 年。AutoManual：大语言模型智能体通过交互式环境学习构建说明手册。*arXiv* 预印本 *arXiv:2405.16247*。第4版。

Wei Fang、Yang Zhang、Kaizhi Qian、James Glass 和 Yada Zhu。2025 年。Play2Prompt：通过工具操作为大语言模型智能体进行零样本工具指令优化。*arXiv* 预印本 *arXiv:2503.14432*。第2版。

Xiangyi Li、Yimin Liu、Wenbo Chen、Bingran You、Zonglin Di、Yifeng He、Shenghan Zheng 等。2026年。Skillsbench：评测智能体技能在多样化任务中的表现。*arXiv* 预印本 *arXiv:2602.12670*。第4版。

Yuxuan Liu、Haoran Li、Yuhao Zhang、Jiahe Guo、Hongyu Luo、Wenbin Hu、Huihao Jing、Kawai Chung、Junle Chen、Changxuan Fan、Qing Zong、Lingyun Xie 和 Yangqiu Song。2026a。Skillscript-bench：评测超越 Markdown 的可执行智能体技能包的自我进化。*arXiv* 预印本 *arXiv:2610.04008*。

Yuxuan Liu、Zhaochen Su、Hangyu Guo、Shijue Huang、Zhixiong Zhang、Kawai Chung、Zhongwei Xie、Tianshi Zheng、Wei Fan、Qing Zong、Haochen Shi、Baixuan Xu、Haoran Li 和 Yangqiu Song。2026b。SkillHorizon：通过技能组合构建长时程智能体任务。*arXiv* 预印本 *arXiv:2610.14966*。

Yuxuan Liu、Zhaochen Su、Lingyun Xie、Yuhao Zhang、Qing Zong、Jiahe Guo、Zhongwei Xie、Yiyan Ji、Yauwai Yim、Hongyu Luo、Xiyu Ren、Chenyu Ruan、Haoran Li 和 Yangqiu Song。2026 年。SkillRevise：通过轨迹条件化技能修订改进大语言模型编写的智能体技能。*arXiv* 预印本 *arXiv:2606.01139*。第4版。

Yuxuan Liu、Zhaochen Su、Yuhao Zhang、Jiahe Guo、Zhongwei Xie、Huihao Jing、Lingyun Xie、Qing Zong、Yauwai Yim、Zhixiong Zhang、Haoran Li 和 Yangqiu Song。2026 年。重新思考自我演化的智能体技能：多轮反馈动力学。*arXiv* 预印本 *arXiv:2608.02636*。

You Lu、Kun Zhang、Bihuan Chen 和 Xin Peng。2026年。DocsChisel：面向大语言模型智能体的自适应工具文档优化框架。*arXiv* 预印本 *arXiv:2608.10037*。第1版。

Aman Madaan、Niket Tandon、Prakhar Gupta、Skyler Hallinan、Luyu Gao、Sarah Wiegreffe、Uri Alon、Nouha Dziri、Shrimai Prabhumoye、Yiming Yang、Shashank Gupta、Bodhisattwa Prasad Majumder、Katherine Hermann、Sean Welleck、Amir Yazdan-bakhsh 和 Peter Clark。2023年。Self-Refine：基于自反馈的迭代式细化。载于神经信息处理系统进展，第36卷。

Cheng Qian、Chi Han、Yi Fung、Yujia Qin、Zhiyuan Liu 和 Heng Ji。2023年。CREATOR：用于解耦大语言模型抽象推理与具体推理的工具创建。载于计算语言学协会：*EMNLP 2023* 论文集，第6922–6939页。计算语言学协会。

Changle Qu、Sunhao Dai、Xiaochi Wei、Hengyi Cai、Shuaiqiang Wang、Dawei Yin、Jun Xu 和 Ji-Rong Wen。2025 年。从探索到掌握：通过自驱动交互使 LLM 掌握工具。载于国际学习表征会议。

Noah Shinn、Federico Cassano、Ashwin Gopinath、Karthik Narasimhan 和 Shunyu Yao。2023年。Re-flexion：借助语言强化学习的语言智能体。载于《神经信息处理系统进展》，第36卷。

Guanzhi Wang、Yuqi Xie、Yunfan Jiang、Ajay Mandlekar、Chaowei Xiao、Yuke Zhu、Linxi Fan 和 Anima Anandkumar。2023年。Voyager：一种借助大语言模型的开放式具身智能体。*arXiv* 预印本 *arXiv:2305.16291*。

Yifan Yang、Ziyang Gong、Weiquan Huang、Qihao Yang、Ziwei Zhou、Zisu Huang、Yan Li、Xuemei Gao、Qi Dai、Bei Liu、Kai Qiu、Yuqing Yang、Dongdong Chen、Xue Yang 和 ChongLuo。2026年。SkillOpt：自我演化的智能体技能的执行策略。*arXiv* 预印本 *arXiv:2605.23904*。第2版。

Shunyu Yao、NoahShinn、Pedram Razavi 和 Karthik Narasimhan。2024年。 τ-bench：现实世界领域中工具-智能体-用户交互的基准测试。*arXiv* 预印本 *arXiv:2406.12045*。第1版。

Barry Zhang、Keith Lazuka 和 Mahesh Murag。2025年。通过智能体技能让智能体适应现实世界。Anthropic Engineering。

Genrui Zhang、Erle Zhu、Jinfeng Zhou、Caiyan Jia 和 Hongning Wang。2026a。SkillEvolver：将技能学习作为元技能。*arXiv* 预印本 *arXiv:2605.10500*。第1版。

Hanrong Zhang、Shicheng Fan、HenryPeng Zou、Yankai Chen、Zhenting Wang、Jiayu Zhou、Chengze Li、Wei-Chieh Huang、Yifei Yao、Kening Zheng、Xue Liu、Xiaoxiao Li 和 Philip S. Yu。2026年b。 CoEvoSkills：通过协同演化验证实现自我演化的智能体技能。 *arXiv* 预印本 *arXiv:2604.01687*。第3版。

Shanshan Zhong、Yi Lu、Jingjie Ning、Yibing Wan、Lihan Feng、Yuyi Ao、Leonardo F. R. Ribeiro、Markus Dreyer、Sean Ammirati 和 Chenyan Xiong。2026年。SkillLearnBench：面向现实世界任务的智能体技能生成持续学习方法基准测试。 *arXiv* 预印本 *arXiv:2604.20087*。第1版。

Yang Zhou、Zihan Dong、Zhenting Wang、Can Jin、Shiyu Zhao、Bangwei Guo、Difei Gu、Linjun Zhang、Mu Zhou 和 Dimitris N. Metaxas。2026年 证据胜于计划：用于技能蒸馏的在线轨迹验证。*arXiv* 预印本 *arXiv:2605.09192*。第 2 版。

诊断研究：数据与分析

方法	原始任务				任务变体			
	评估1	评估2	Mean	|Δ|	表征	实例	要求	总体
共享参考								
无技能	45.74	37.66	41.70	8.08	40.56	44.66	38.50	41.09
Skill-Creator	42.15	33.24	37.70	8.91	47.96	39.44	37.58	41.76
进化技能								
SkillEvolver	63.22	62.34	62.78	0.87	53.89	59.56	48.65	53.81
SPARK	36.63	40.93	38.78	4.30	44.05	39.77	42.66	42.26
CoEvoSkills	52.44	46.44	49.44	6.00	38.46	35.49	35.71	36.60
Self-Refine	60.38	60.54	60.46	0.16	60.21	63.08	52.92	58.55
SkillLearn	54.43	55.80	55.11	1.36	64.50	58.88	50.14	57.80
Trace Evo	63.76	59.89	61.82	3.87	62.11	64.39	50.69	58.84
SkillRevise	57.71	57.43	57.57	0.28	58.27	54.88	50.01	54.36

表4：原始任务评估和变体类别的详细平均奖励（0–100）。 |Δ| 是评估1与评估2之间的绝对差值，按四舍五入前的数值计算。原始任务均值对两次评估赋予相同权重。

原始任务评估。 表4 列出了各项评估分数和类别级变体结果。表 1对 SkillE-volver（81 个任务）、SPARK（84 个任务）、CoEvoSkills（85 个任务）以及 Self-Refine、SkillLearn 和 Trace Evo（各 86 个任务）进行了两次评估并取平均值。Skill-Creator 在相同的 86 个任务上进行了两次评估并取平均值。SkillRevise 在两次评估中分别使用了 86 个和 83 个任务。部分奖励保留其原始值，而预算耗尽的执行得分为零。所有行均报告 0–100 刻度上的平均奖励。

无技能参照。 所显示的原始任务无技能参照使用了 CoEvoSkills 的两次85任务评估。

A.1 任务级增益的来源与复现

每种方法的两次评估中，任务身份均一一对应：SkillEvolver 有 81 个，CoEvoSkills 有 85 个。每次评估都将演化技能条件与同一任务的无技能执行进行配对。对于 0–100 量表上的奖励差值 d_i，两项贡献分别为 $G = \sum_i \max(d_i,\ 0)/N$ 和 $L = \sum_i \max(-d_i,\ 0)/N$；两者之差即为配对平均增益。所有 N 个任务（包括平局）都计入分母。

度量	运行1	运行2
无技能奖励	37.60	37.32
进化后奖励	63.22	62.34
正向贡献 G	30.91	28.32
负向贡献 L	5.30	3.30
净增益	25.61	25.02
改进任务	27	26
平局任务	48	52
退步任务	6	3

表5：两次 SkillEvolver 评估的任务级增益分解。

在至少一次评估中有所改进的 36 个 SkillEvolver 任务中，有 17 个在两次评估中均有所改进；两次评估的重叠数为 17/36，而首次评估中有所改进的任务，其改进保持率为 17/27。CoEvoSkills 在两次评估中分别有 17 个和 18 个改进任务，其中有 10 个任务在两次评估中均有所改进；任一评估中有所改进的任务共有 25 个。表 6 展示了两次评估之间的变化。CoEvoSkills 的第一次/第二次正向贡献分别为 17.14/19.44，负向贡献分别为 6.90/9.48，净增益分别为 10.24/9.96。

	SkillEvolver			CoEvoSkills		
评估1	I	T	R	I	T	R
改进	17	10	0	10	7	0
Tied	7	38	3	8	47	4
退步	2	4	0	0	3	6

表6：从评估1（行）到评估2（列）的任务级收益转变。I/T/R 分别表示相对于各评估配对的无技能结果，改进、持平和退步。

执行结果的变化。Self-Refine 在每次评估中有 50 项任务获得完整奖励，其中 9 项任务失去完整奖励，另有 9 项获得完整奖励。SkillLearn 获得完整奖励的任务数从 44 项变为 46 项，其中 7 项失去完整奖励，9 项获得完整奖励；Trace Evo 的数量则从 53 项变为 48 项，其中 10 项失去完整奖励，5 项获得完整奖励。

A.2 任务变体构建

这组包含 233 个变体的数据集源自 SkillsBench 的 81 项原始任务：81 个表征变体（P1）、71 个实例变体（P2）以及 81 个耦合需求变体（P3）。每个变体都保留其源任务标识，并包含一项公开请求、任务资源和环境配置，以及相应的评估规范。这三类变体区分的是任务预期发生的变化，而非为每个领域规定统一的变换方式。

SkillHorizon（Liu 等人，2026b）研究了一种互补的任务构建场景，将可复用的技能流程组合成由相互依赖活动构成的长时程任务。在此，变体会修改现有 SkillsBench 任务的要求。

表征变更（P1）。这些变体会改变操作接收输入或呈现输出的方式，包括路径、字段名称和序列化约定。它们测试技能是否能够从变更后的表征中恢复所需操作，而不是依赖原始的表面形式。

实例变更（P2）。这些变体会替换工作流所处理的具体输入或关系，例如数据记录或授权关系。例如，依赖项审计变体会修改所提供锁定文件中的软件包位置和依赖项元数据。任务仍然要求执行依赖项审计，但工作流必须从新的实例中推导其发现结果。

耦合需求变更（P3）。这些变体会添加或修改必须共同满足的要求。在 3D 扫描任务中，原始请求会选出最大的连通分量，并根据体积和材料密度计算其质量。其 P3 变体仅在满足以下条件的分量中选择体积最大的分量：该分量的材料存在于所提供的密度表中，且其包围盒中心的 X 坐标位于 $[40, 60]$ 厘米处。输出仍保留质量和材料 ID 字段，但现在必须先进行资格筛选，然后才能选择分量。

打包与评估。 变体请求会指定适用的输入与输出契约，并在需要时提供共同说明。公开任务资源文件与私有评分资源分别打包。技能取自相应的原始 SkillsBench 任务：为 Skill-Creator 提供生成的技能，为演化方法提供进化技能。它们在变体评估期间保持冻结。无技能不提供任何技能。每次执行都使用相应变体的评分器进行评分。

A.3 按任务变更类别划分的效用

表7 报告了各变体在 0–100 分制下，使用技能与无技能执行之间的平均奖励差异，并采用各方法配对任务队列中的任务。Skill-Creator、Self-Refine、SkillLearn、Trace Evo 和 SkillRevise 各覆盖 233 个变体（P1/P2/P3 分别为 81/71/81）。SkillEvolver 覆盖 222 个（77/68/77），CoEvoSkills 和 SPARK 则各覆盖 221 个（77/68/76）。SkillEvolver 和 CoEvoSkills 配对后的无技能均值分别为 41.87 和 41.61。SPARK 的总分按任务数量对类别均值加权。

方法	P1	P2	P3
Skill-Creator	+7.40	−5.22	−0.92
SkillEvolver	+11.85	+14.56	+9.72
SPARK	+2.00	−5.23	+4.53
CoEvoSkills	−3.59	−9.51	−2.42
Self-Refine	+19.66	+18.42	+14.41
SkillLearn	+23.95	+14.23	+11.64
Trace Evo	+21.56	+19.73	+12.19
SkillRevise	+17.71	+10.22	+11.51

表 7：按任务变化类别划分的配对奖励增益。

B 操作知识与修订详情

本附录详细说明第 4 节中介绍的脚本准备、知识表征、反馈构建和修订接口。

准备可执行操作。脚本构建从公开任务请求和可用的辅助程序实现开始。任务族审查会识别可复用的操作、其输入和输出接口、所需依赖项，以及仍由智能体决定的事项。适用的辅助程序会被复用；缺失的操作则实现为参数化函数和命令行接口。生成的软件包将实现文件与合成单元测试分开，并包含使用示例和声明的限制。实例特定答案、固定任务路径和隐藏测试内容不会纳入生成输入或输出。

生成提示要求测试至少覆盖两个不同的合成输入和一个无效输入。组件测试运行器会执行这些测试

在网络禁用容器中运行，并以只读方式挂载软件包。这些检查会测试辅助接口；下游任务评估仍是一个独立阶段。脚本集准备就绪后 P，一次编写调用会根据 M_0 中的公开请求和实际接口来编写 P。后续的指导修订会保留 P。附录 E 提供了这两种提示模板。

基于源的知识记录。 提取输入保留所提供公共脚本的完整文本，并附有编号的源代码行。对于可解析的 Python 脚本，AST 索引记录定义、参数、调用表达式、条件分支及其源代码跨度；其他源则保留其原始文本。模型根据方程 1生成最多 12 条操作性知识记录。每个源引用都指定一个文件和包含首尾的行范围。系统会检查所提供源中是否存在每个范围，并附加该位置的确切代码及源哈希。

工作流比较与反馈。 给定$\mathcal{K}$、任务请求、当前指导和历史轨迹后，单次模型调用会生成工作流比较及相关建议。每条建议都与一个或多个提出行动的比较相关联；这些比较中的知识引用共同涵盖建议所引用的知识引用。输出最多包含 12 行比较和 12 条反馈项。反馈会引用准确的任务请求引文；对于相互冲突的指令，则引用当前指导引文。验证会检查知识引用、比较与反馈的关联关系以及所引用的文本。附录E 规定了输出字段。

针对性编辑与候选项形成。 修订模型会对指定的 Markdown 文件返回无修改结果，或最多四处互不重叠的编辑。每处编辑都需指定文件、与原文完全一致的段落、不超过 8,000 个字符的替换文本，以及一个或多个反馈标识符。系统会检查路径是否可编辑、原文段落是否唯一匹配、反馈引用是否有效，以及编辑范围是否互不重叠。每个修改过的文件都会保留部分原始非空文本；脚本及其他文件保持不变。有效编辑会应用于形成候选项。若反馈为空、响应表示不作修改，或修订输出无效，则保留当前指导。候选项会在下游执行和任务验证器评估之前冻结；验证器结果不会用于选择其编辑内容。

C 主要评估：协议与覆盖范围

任务与评估。 评估包含 84 项 SkillsBench 任务和 99 项 SkillLearnBench 任务。候选项单元可以是特定于任务的，也可以由使用相同技能包的多个任务共享。每个任务–方法–条件组合都有三条固定的评估记录，候选项在评估前冻结。

原生工作流。 SkillLearn 和 SkillRevise 各自运行一次原生工作流，最多进行一次修订。随后选定并冻结候选项，再于独立会话和干净工作区中进行三次下游执行。每种方法的原生结果涵盖 83 项 SkillsBench 任务和 99 项 SkillLearnBench 任务。

验证器反馈。 报告包含父级判断以及最多四条记录的测试失败。保留可用的观测结果；缺失的预期值或观测值仍保持未指定。诊断包含父级判断、有界的原生结果元数据，以及最多八条带有诊断消息和跟踪信息的失败记录；报告长度上限为 8,192 个 token。两种报告均不包含验证器源代码和隐藏测试源代码。

修订接口与停止策略。 初始指导依据公开请求和脚本实现生成，不包含人工编写的指导。所有条件均接收相同的公开源投影和未标注的父级执行轨迹。Self-Refine 使用一次修订调用；TraceEvo、SkillLearn 和 SkillRevise 则先进行分析调用，再进行修订调用。SkillLearn 和 SkillRevise 均调整为一次由反馈条件驱动的更新。没有任何方法会跨任务更新记忆，SkillRevise 的原则库也保持冻结。Binary、Report 和 Diagnostic 在记录到父级通过时保留初始技能，仅在失败时进行修订。无验证器和 VISE 不依据任务验证器判定进行修订，也不采用这一基于验证器的停止规则。

模型与执行设置。 在受控反馈比较中，修订调用使用 gpt-5.5，温度设为零，每次调用的生成上限为 16,384 个词元。下游执行同样使用 gpt-5.5，并在各条件下保留每个任务已登记的执行限制。

条件	输入技能	更新	下游评估
Skill-Creator	已生成 $S_0 = (P,\ M_0)$	无后续演化	S_0的三次执行
无验证器	初始 S_0	使用公开请求、脚本和无标签轨迹的方法特定更新	冻结输出；执行三次
二元 / 报告 / 诊断	初始 S_0	使用指定任务的方法特定更新-验证器反馈；在父级通过时保留 S_0	冻结输出；执行三次
VISE	实际传入的演化结果 $S = (P,\ M)$	从 P 中提取知识，并将其与 M 对齐，并使用工具支撑的反馈修订 M	冻结 $S' = (P,\ M')$；执行三次
原生	方法自身的工作流中-put	运行原始版本 SkillLearn 或 SkillRevise 工作流，最多进行一次修订	冻结选定的输出；执行三次

表 8：主要评估中的起始技能和更新路径。

指标。 非 VISE 比较包含十六个方法–反馈单元，以及每个任务共享的一个 Skill-Creator 基线，每个单元均执行三次。对于具有 c 次成功执行次数的任务，P@1 取平均值 $c/3$，P@3 取平均值 $\mathbf{1}[c \geq 1]$，Hit[3]取平均值 $\mathbf{1}[c = 3]$。对于四种非 VISE 反馈条件和 Skill-Creator，三个指标均使用相同的 183 个任务（84 个 SkillsBench 任务和 99 个 SkillLearnBench 任务）；VISE 使用下文所述的完整得分子集。总体上对任务赋予相同权重，而不是对各基准测试的百分比取平均值。

起始技能和更新路径。 表 8 区分了初始技能构建与后续修订的输入。VISE 将前一演化阶段的冻结输出作为其输入技能$S = （P,\ M）$。历史轨迹 τ提供执行上下文。

计算成本。 表9 比较了技能修订所记录的令牌使用量和累计 API 时间。无验证器、二元、报告和诊断度量从 S_0开始的更新；VISE 度量传入的演化技能的额外优化$S = （P,\ M）$。每一行均对相同的 122 个任务取平均值，其中包括保留输入技能的情况。VISE 为每项技能使用 41.62k–44.79k 个输入令牌和 1.70k–1.95k 个输出令牌，API 时间范围为 57.97 至 82.27 秒。

组件比较协议。 表3 使用 60 个任务的 TraceEvo 子集（每个基准测试包含 30 个任务），每个任务和条件对应三条记录。无工作流修订使用表 2 中这 60 个任务对应的无验证器记录，不进行后续的 VISE 修订。无结构化反馈在使用源代码和无标签轨迹的基础上额外进行一次修订，但不使用结构化知识或工作流对齐。完整的 VISE 则将实现细节组织为

方法	反馈	输入 (k)	输出 (k)	时间 (s)
Self-Refine	无验证器	20.38	1.92	36.97
	二元	14.25	1.58	26.36
	报告	15.17	1.37	32.58
	诊断	15.21	1.42	26.33
	VISE	42.33	1.85	82.27
TraceEvo	无验证器	41.02	2.98	61.70
	二元	28.41	2.03	40.26
	报告	30.32	2.00	42.18
	诊断	30.26	1.91	41.94
	VISE	41.62	1.70	58.55
SkillLearn	无验证器	37.57	2.68	60.61
	二元	26.53	1.93	37.25
	报告	25.45	1.70	37.47
	诊断	28.41	1.73	38.96
	VISE	44.79	1.95	64.85
SkillRevise	无验证器	41.33	2.66	36.92
	二元	28.91	1.88	24.05
	报告	29.74	2.06	28.38
	诊断	30.70	2.06	29.23
	VISE	42.79	1.83	57.97

表9：122项具有完整成本记录的共享任务的平均修订阶段令牌使用量和累计API时间。

操作知识，并构建反馈，明确说明应如何修订现有指导。无工作流对齐条件复用提取出的知识，但移除动作绑定记录和发现绑定，同时保留通用的证据源契约和修订验证器。

组件成功次数分布。表10 将表3 中的分数展开为任务数量。令n_j 表示在三条评估记录中具有j 次成功的任务数量。在至少成功一次的 $60 - n_0$ 项任务中，平均成功次数为($n_1 + 2n_2 + 3n_3)/(60 - n_0)$。

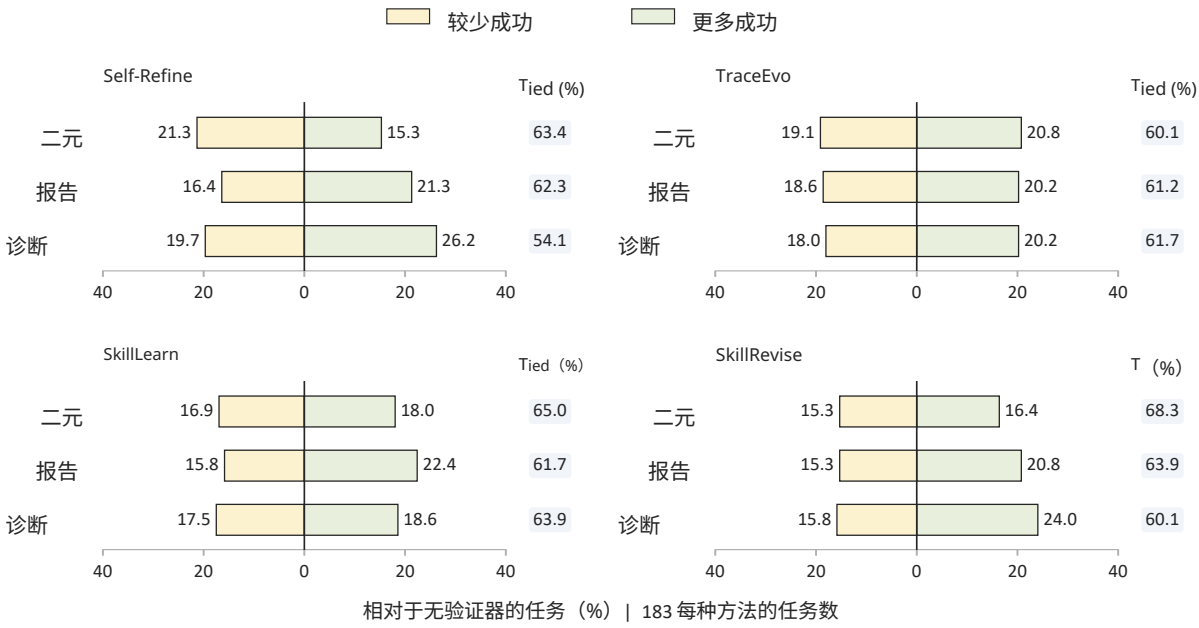

图7：在每种方法的 183 项任务中，相对于无验证器的任务级变化。条形图显示了三次评估中成功次数较少（左侧）或较多（右侧）的百分比；侧栏显示持平的情况。

条件	n_0	n_1	n_2	n_3	Mean
无工作流修订	14	18	13	15	1.93
无结构化反馈	24	8	13	15	2.19
无工作流对齐	17	18	8	17	1.98
原始来源修订	22	14	16	8	1.84
完整 VISE	14	6	22	18	2.26

表10：TraceEvo 组件比较中 60 项任务的成功次数分布。均值指至少成功一次的任务中的成功次数。

评估覆盖范围。对于每种方法，所有三个 VISE 指标均使用同一组任务，这些任务的初始技能和 VISE 修订版均具有完整的评估记录。表11 报告了每个基准测试所用的任务数量。

方法	SkillsBench	SkillLearnBench	总体
Self-Refine	82/84	97/99	179/183
TraceEvo	82/84	97/99	179/183
SkillLearn	81/84	96/99	177/183
SkillRevise	82/84	97/99	179/183

表 11：VISE 评估覆盖率：对于初始技能及其修订版本，均具有三个原生得分的任务，相对于完整任务集的比例。

D 验证器反馈的任务级影响

图7 将比较分解为成功执行次数多于、等于或少于无验证器的任务。每次比较均使用相同的 183 个任务，并针对每个条件进行三次评估。

类别级比较。 图4采用 SkillsBench 的官方主要任务类别（Li 等人，2026年）以及 SkillLearnBench 的官方任务族类别（Zhong 等人，2026年）。对于每种方法，五个条件均使用与图 6相同的完整任务集：Self-Refine、TraceEvo、SkillLearn 和 Skill- Revise 分别包含 179、179、177 和 179 个任务。每个单元格均以反馈条件在同一类别任务上的 P@1 减去表 2中的无验证器 P@1，其中也包括 VISE 比较。

成功次数转变。 图8 比较了每项任务在无验证器和 VISE 下的成功次数。对于 VISE 下分别新达到三次成功的 33、37、36 和 35 项任务，其中分别有 28、25、23 和 22 项在无验证器下成功过一两次。反过来，分别有 7、9、3 和 14 项任务从三次成功降至较少成功。因此，大多数新观察到的三次成功任务来自间歇性成功组，而非零成功组。方法依次为 Self-Refine、TraceEvo、SkillLearn 和 SkillRevise；下方两幅图采用

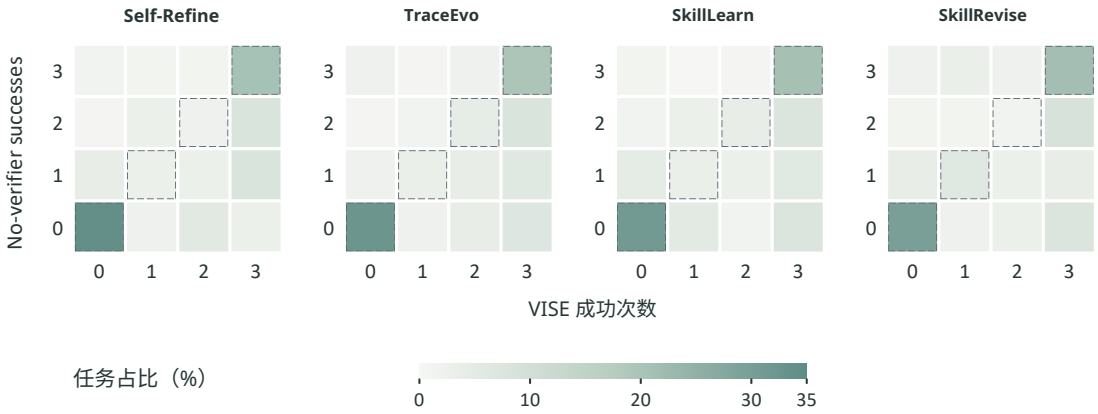

图8：从无验证器到 VISE 的任务级成功次数转变。颜色表示每一对成功次数中任务的百分比，不同方法采用共享刻度。虚线对角单元格表示成功次数未发生变化；其右侧单元格表示 VISE 下成功次数更多，左侧单元格表示成功次数更少。各队列分别包含 179、179、177 和 179 个任务。

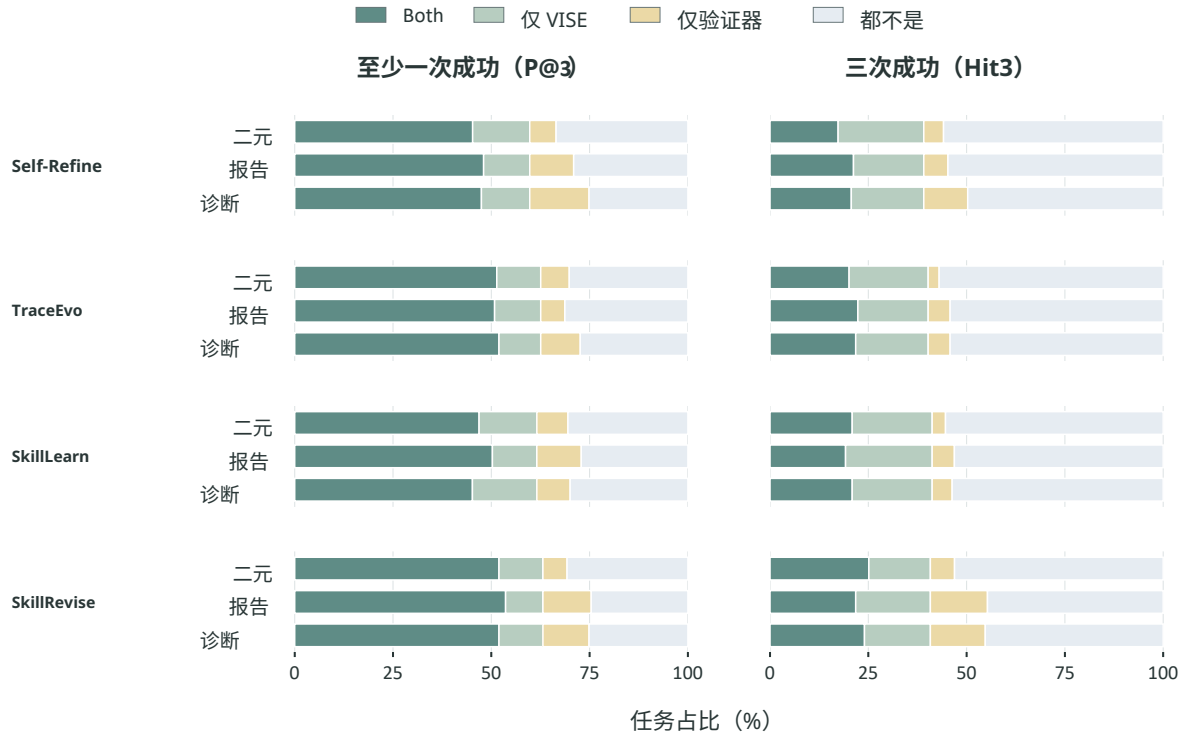

图9：VISE 与任务验证器反馈之间的成功任务重叠。每个条形图将 VISE 与其标注的反馈条件在相同任务上进行比较。四个分段根据以下情况对任务进行划分：两者均满足、仅 VISE 满足、仅验证器条件满足，或两者均未满足面板的成功标准。左：至少一次成功；右：三次成功。队列与图8相同。

与类别分析相同的各方法内队列。

成功任务重叠。 图 9比较了 VISE 与每个任务验证器反馈条件所解决任务的身份。在三次成功标准下，在全部十二项比较中，仅由 VISE 完成的任务数都多于仅由验证器完成的任务数。

对于至少成功一次的情况，排序可能有所不同：在 Self-Refine 与 Diagnostic 的比较中，只有 VISE 成功的任务有 22 个，只有 Diagnostic 成功的任务有 27 个；而对应的三次成功数量分别为 33 个和 20 个。因此，在同一比较中，任务覆盖率和重复成功区分出的是不同的任务集合。

E 提示模板

以下方框展示了已实现提示接口的精简形式。任务请求、源文件、指导和跟踪信息以输入占位符的形式出现；输出字段名称遵循具体实现。这些模板遵循一条数据规则：嵌入的源文件、请求、指导和观测结果均视为数据，而非指令；不得臆造任务结果或隐藏答案。

构建缺失脚本

指令。 为提供的公共工作流实现可复用辅助函数。通过参数化函数和命令行接口公开明确的输入与输出。保留原始输入，并报告不受支持的情况。使任务特定值可配置；不要将答案、隐藏测试、固定任务路径或任务资源嵌入软件包中。将实现与合成测试分别生成，不要放入 SKILL.md。**输入。** 公共任务族请求和操作审查，包括拟议的接口、依赖项和限制。**输出。** 返回一个包含名称、描述、依赖项、文件、用法_示例和限制的 JSON 软件包。每个文件都包含路径和内容。将实现放在 scripts/ 下，将合成单元测试放在 tests/ 下。至少覆盖两种不同的输入和一种无效输入。记录命令行界面用法。不得执行网络访问、调用 Shell 命令、读取凭据、安装软件包，或在辅助函数中封装 LLM 调用。

初始工作流指南

指令。 为可执行技能包编写初始可复用技能文档。使用任务请求作为预期用法的示例。描述通用程序以及只读脚本所暴露的实际接口；不要解决任务或嵌入其答案。编写简洁的指导，帮助未来的智能体决定何时以及如何使用这些脚本。参考所提供源代码支持的路径、符号、参数、输出和工作流。

输入。 [CREATOR_规范], [_], [_], 创作适配器任务请求授权目标和公开脚本]。每个目标都指定一个 Markdown 路径和所需技能名称。

输出。 调用 submit_initial_skill_package，并传入摘要和文档。每个文档包含路径和内容。每个已授权文档恰好返回一次，并保留其路径和所需名称。每个文档以包含 name 和 description 的 YAML 前置元数据开头。脚本保持不变。

操作知识提取

指令。 在诊断任务之前，先从所提供的脚本中提取可复用的操作知识。描述各项能力、使用时机、有序步骤以及条件限制。结合上下文阅读实现，而不仅仅是名称或语法节点。将相关操作归组为工作流，而不是为每个节点生成一条记录。可用的辅助函数未必是必需的，默认值也未必是错误的。**输入。** [SOURCE_PACKET]包含公开脚本路径、带编号的源代码行，以及在支持解析时的语法索引。其他来源保留其文本。**输出。** 返回一个 JSON 对象，其中包含最多 12 条记录的工作流数组；每条记录包含 id、能力、适用性、步骤、约束和来源。每个源引用包含 path、start_line 和 end_line。使用所提供的行号选择狭窄且包含端点的范围；宿主会附加其精确的源代码文本。允许空数组。根据所引用的实现推断适用性和使用建议。

以工作流为依据的反馈

说明。 根据提供的指导，对齐公共工作流与所有获准的 Markdown 文件。按工作流整理比较结果。当存在相关限制时，区分现有操作、已知限制和未完成的过渡。不同文件中的等效操作视为已涵盖；仅说明限制并不等于提供替代路径。应根据适用条件，建议必要的剩余操作，而不是重复陈述或进行风格上的修改。保留有用的指导。**输入。** [TASK_REQUEST], [SKILL_MDS],[OPERATIONAL_KNOWLEDGE], 和 [HISTORICAL_TRACE]。SKILL_MDS 将获准路径映射到其当前文本。**输出。** 返回一个包含比较结果的 JSON 对象。每项比较包含对齐行，以及一项反馈或 null。对齐行总数为 1–12 行，反馈项最多 12 项。**对齐字段。** requirement、existing_operation、existing_paths、known_limitation、knowledge_ids、remaining_transition 和 decision。可选值为 COVERED、ADD_OPERATION、ADD_FALLBACK、CORRECT_OPERATION 或 UNKNOWN。COVERED 必须以现有操作和获准的 Markdown 路径为依据；缺少此类依据时使用 UNKNOWN。**反馈字段。** id 、 kind 、 knowledge_ids 、 request_quotes 、 skill_quotes、event_ids、issue、recommendation 和 applicability。使用 MISSING_GUIDANCE 或 CONFLICTING_GUIDANCE。每项都必须引用所提供的知识和请求中的准确引文；若存在冲突，还须引用所提供 Markdown 中的准确段落。将 event_ids 设为空数组。**关联规则。** 每项反馈至少由一条具有剩余过渡且知识标识符重叠的操作行提供支持。其操作行合计必须涵盖该反馈项引用的全部知识标识符。COVERED 和 UNKNOWN 行可以提供背景信息，但不能作为拟议修改的依据。若比较结果不支持有用的修订，则返回 null。

反馈引导的工作流修订

说明。 仅根据提供的条件性反馈改进获准的 Markdown。评估建议并保留有用的现有说明。不要将可选能力变成无条件要求，也不要把提取的全部知识粘贴到技能中。有依据的指导缺口即使没有运行时异常，也足以成为编辑理由。脚本及所有其他文件均为只读。

输入。 [任务_请求], [技能_MDS][操作性_知识], [反馈], 和 [历史_轨迹]。

输出。 返回决策：如果没有有用的更改，则选择 KEEP；否则选择 EDIT 并提供一个 edits 数组。每项编辑包含 path、old、new 和 finding_ids。原始段落必须是其授权文件中唯一且完全匹配的子字符串；finding_ids 用于引用该编辑所针对的反馈。在所有授权文件中，最多返回四项互不重叠的编辑，并确保每个经过编辑的文件都保留部分原有的非空文本。

无结构化反馈

指令。 根据任务、公开源代码脚本和历史执行轨迹，修订技能指导以供将来使用。自行找出具体的改进之处。脚本和资源文件均为只读。保留名称和前置元数据，不要替换整个指导文件。**输入。** [任务_请求], [技能_MDS][公开_脚本]，和 [历史_轨迹],，不含结构化反馈包或任务验证器结果。**输出。** 返回决策：KEEP，或决策：EDIT 最多进行四处互不重叠的修订。每处修订都包含 path、old 和 new。每个原始片段在其文件中恰好出现一次；每个替换内容最多包含 8,000 个字符。

无工作流对齐

Instruction. 将脚本派生的操作知识与公开请求及获授权的 Markdown 进行比较。提供可执行的、有条件的反馈，而不是源内容摘要。指出相关的缺失指令，或与实现相冲突的指令。保留已有的有用指导。**Inputs.** [任务_请求], [技能_Markdown][操作_知识], 以及 [历史_追踪]。**Output.** 直接返回反馈数组，不要包含对齐行或操作与反馈的关联。使用与完整反馈接口相同的反馈字段、知识引用和精确引文要求。使用缺失_指导或冲突_指导，并将事件_标识符置为空。最多返回 12 项；如果没有有用且有依据的差异，则返回空数组。生成的反馈使用相同的工作流修订接口。

无工作流修订直接使用无验证器技能，因此不会引入额外的 VISE 提示。

F 一个完整的工作流修订示例

本附录扩展了第6.3节中的视频物体计数示例，沿着一个 Self-Refine 技能从其初始指导到修订后执行的过程展开。任务要求对视频帧中的物体进行计数，并将计数结果记录在 CSV 文件中。修订前后的可执行计数器完全相同；变化仅限于 `SKILL.md`。

实施证据。
文件 scripts/count_objects.py 提供五个必需参数：–tool、–input_image、–object_image、–threshold 和 –dedup_min_dist（第 25–44 行）。其计数函数以灰度读取两幅图像，应用归一化模板匹配，按阈值筛选匹配项，并删除彼此靠近的重复项（第 47–73 行）。命令行入口会打印最终的整数计数，以及输入和模板路径（第 76–86 行）。

操作知识。
VISE 将这些内容归为
源代码跨度，并记录在 count-objects-cli 中。其能力是基于模板的目标计数；其条件要求输入图像和模板图像均可读取。该记录将输入准备、可配置的匹配条件和输出获取联系起来，形成一个用于获取计数结果的流程，所得结果可供后续工作流使用。

指导差距与反馈。
传入的指导仅要求检查脚本并使用运行时证据。工作流对比识别出缺失的计数过程，并生成 finding-1，该发现与 count-objects-cli 以及任务的对象计数要求相关联。建议添加调用过程，并说明对每个帧–模板对使用打印出的计数。帧提取和 CSV 写入是计数器之外的独立操作。

传入指导：准确摘录

检查 scripts/ 下的文件，以确定其接口和行为。在可用时使用运行时证据。

新增指导：准确摘录

使用模板计数物体
When用于计算样本物体图像在另一张图像中出现的次数，该软件包在 scripts/count_objects.py 中提供了一个计数工具。 . ..
根据图像/模板尺度选择阈值和去重距离，并根据需要进行验证。该命令会打印一行，报告该输入图像和模板对应的对象数量；对每个帧/模板对使用所报告的计数。

接受的修订。
一次编辑将计数过程追加到 object_counter/SKILL.md，并引用 finding-1，同时保留现有说明。数值阈值和帧选择仍属于执行时选择。

在执行中使用。
两次记录的执行都调用了计数器，但帧选择和匹配的参数不同。在传入技能执行中，CSV 文件已经写入后，通过四次调用检查单个帧。在修订后的技能执行中，调用发生在帧–模板循环内：subprocess.check_output 捕获打印结果，正则表达式提取其整数计数，然后将计数汇总为硬币和乌龟，最后由 csv.DictWriter 写入包含 frame_id 的五行记录。修订后的执行使用计数器的输出构建最终 CSV。

G 主要结果任务热图

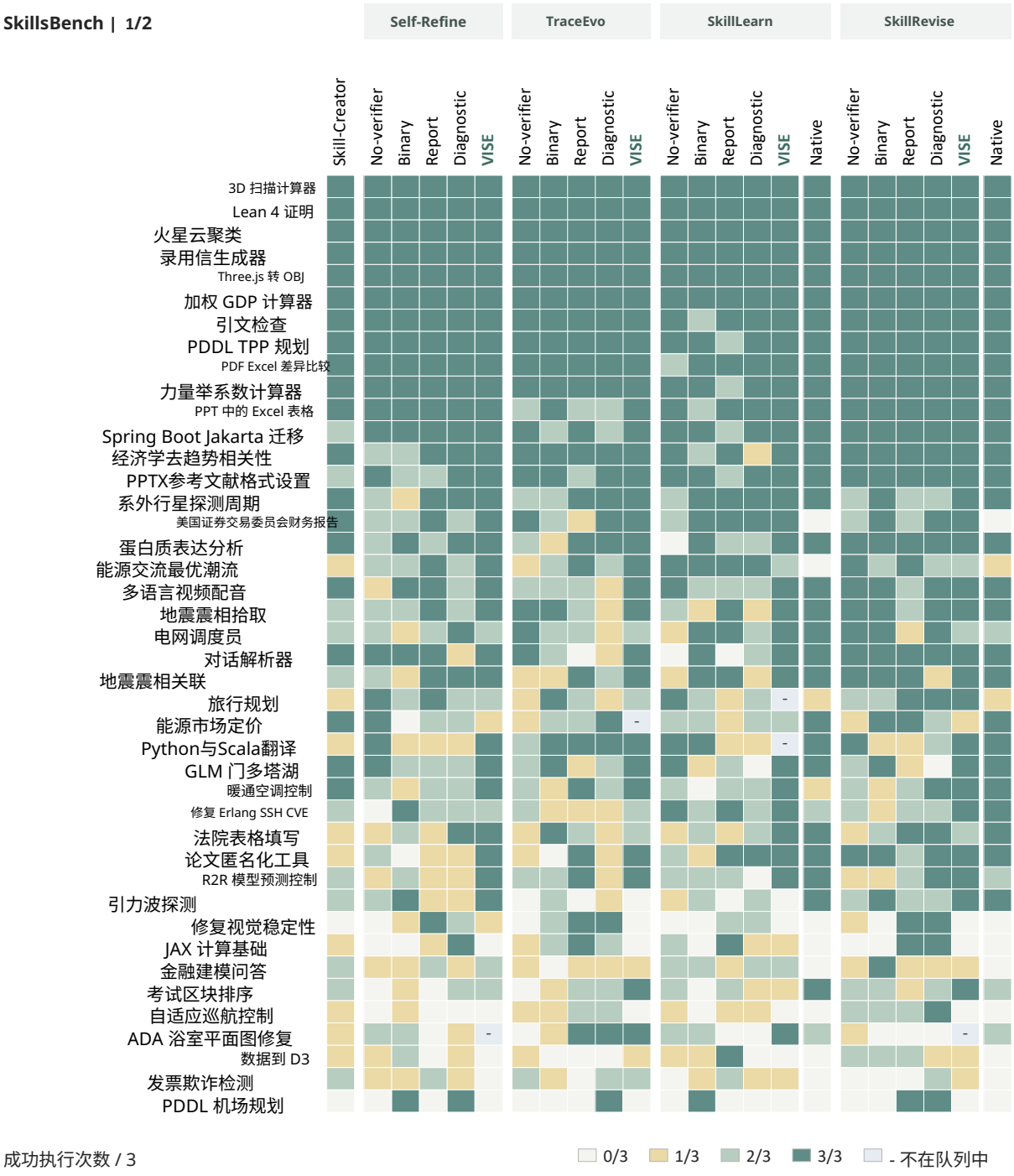

图10：SkillsBench 上的任务级结果（共2部分，第1部分）。每个单元格显示表格中某一条件下三次执行中的成功次数 2；短横线表示任务不属于该条件的评估队列。任务按照 Skill-Creator 和十六个非 VISE 对照条件下的成功总次数排序，各列采用相同的顺序。Native 表示每种方法的原始工作流。

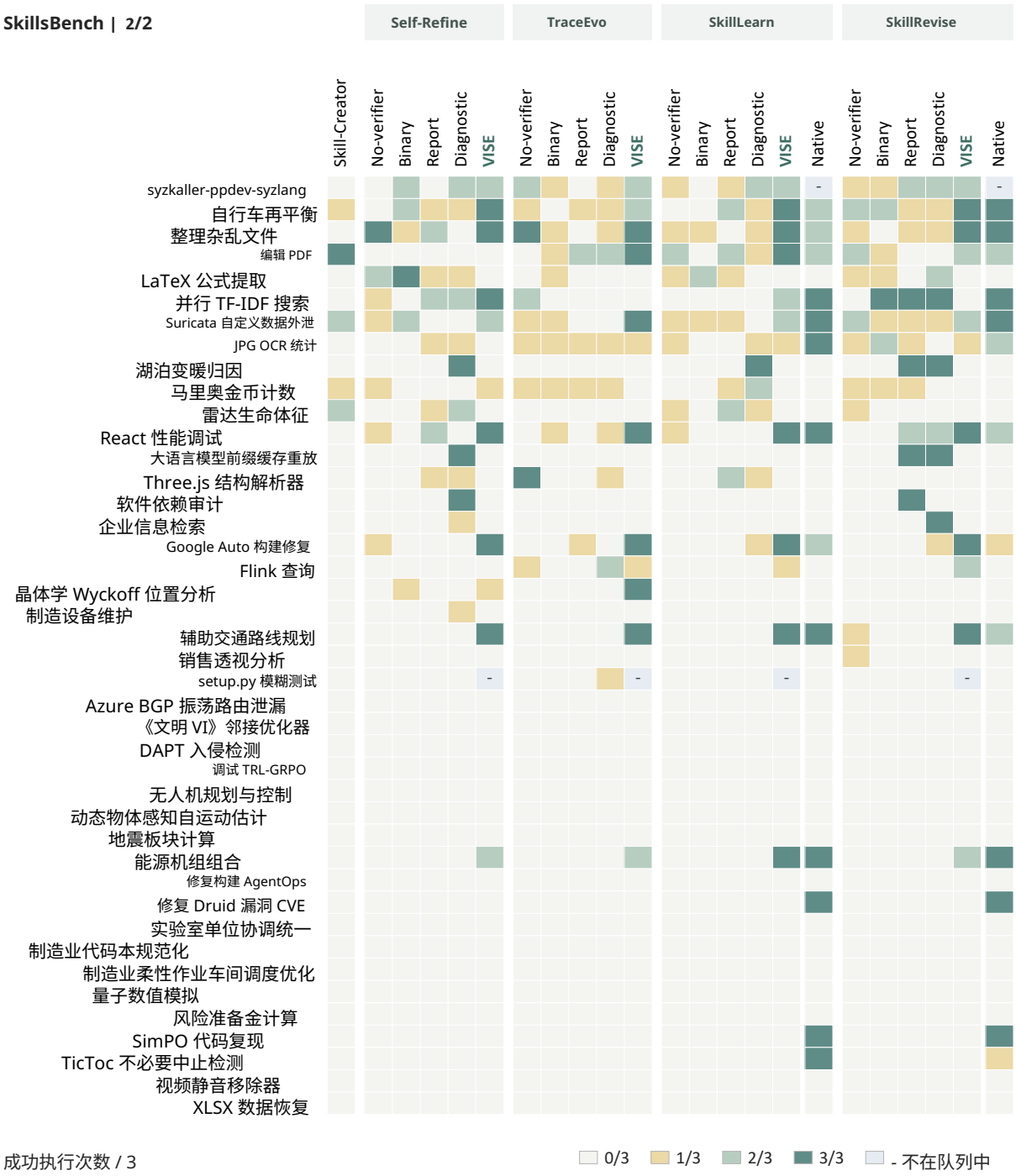

图11：SkillsBench 上的任务级结果（第 2 部分，共 2 部分）。单元格显示三次尝试中的成功执行次数；破折号标记不属于相应评估队列的任务。列和任务顺序遵循图 10。

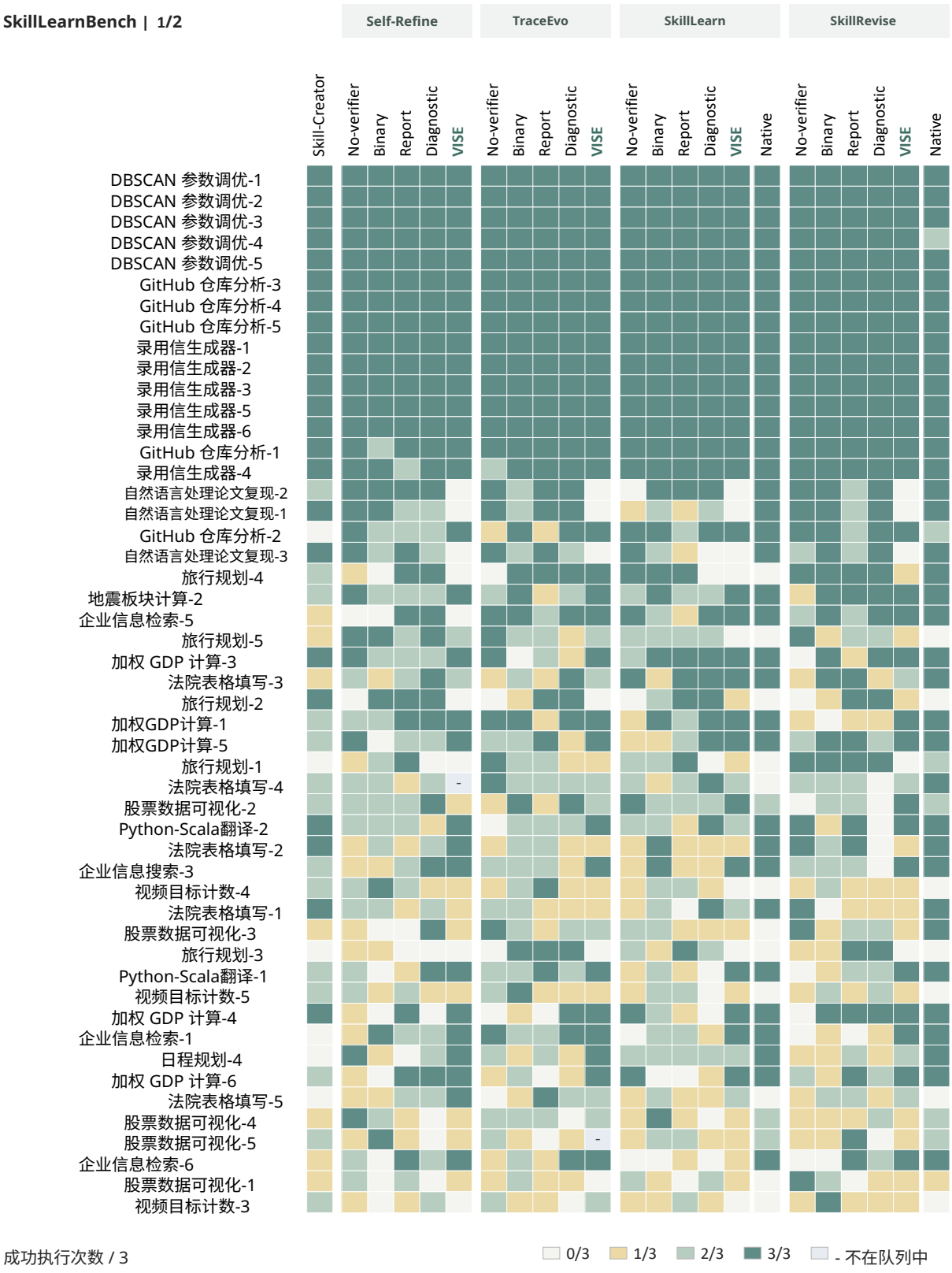

图12：SkillLearnBench 上的任务级结果（第 1 部分，共 2 部分）。单元格显示三次执行中的成功执行次数；短横线表示任务不属于相应的评估队列。任务按照 Skill- CREATOR 和 16 个非 VISE 受控条件下的成功总次数排序，使用与图 10 相同的列。

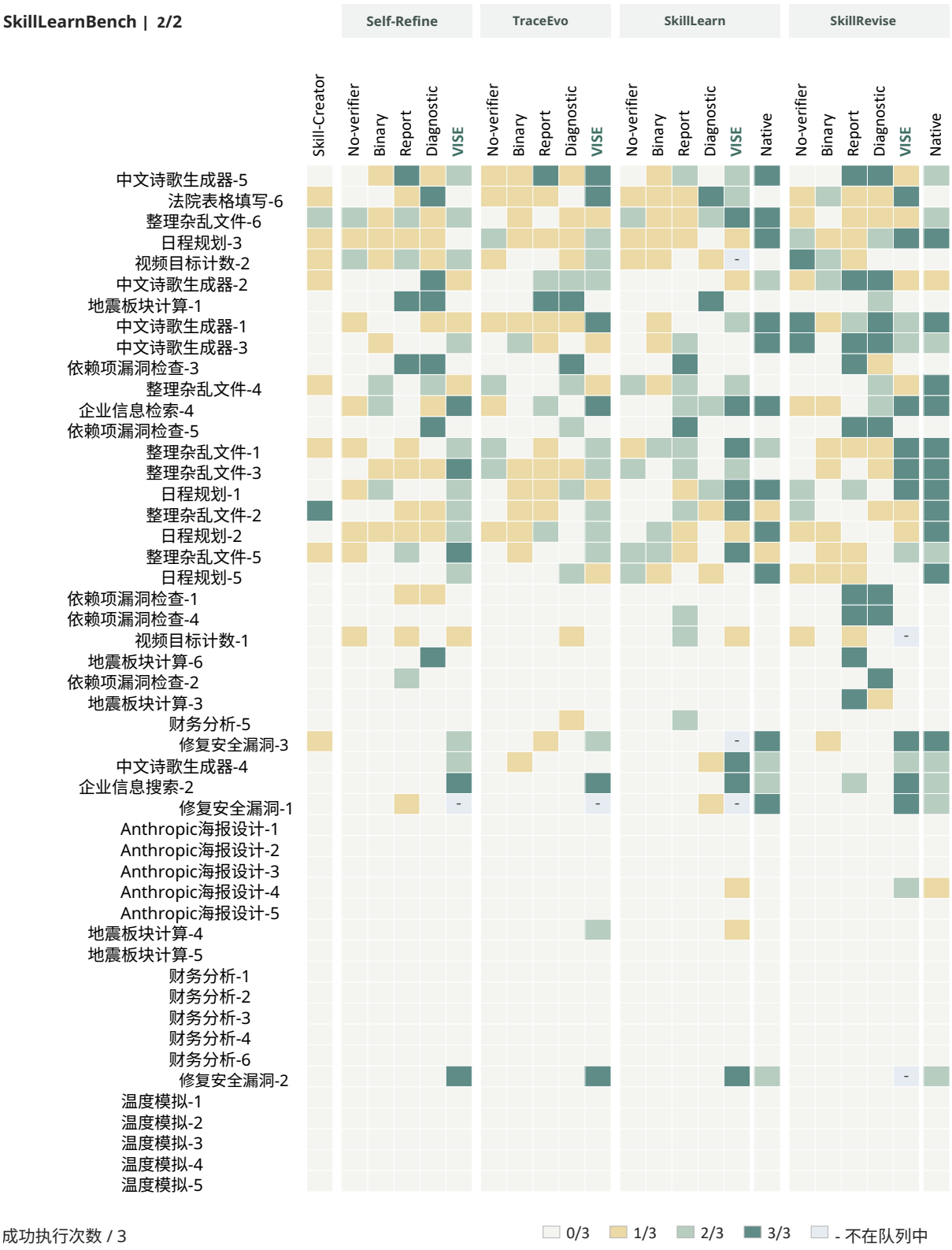

图13：SkillLearnBench 上的任务级结果（第 2 部分，共 2 部分）。单元格显示三次执行中的成功次数；短横线表示不属于相应评估队列的任务。列和任务顺序与图12一致。