上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

SJTU

模型之外的智能

LLM Agent 的外部化范式

周宸宇

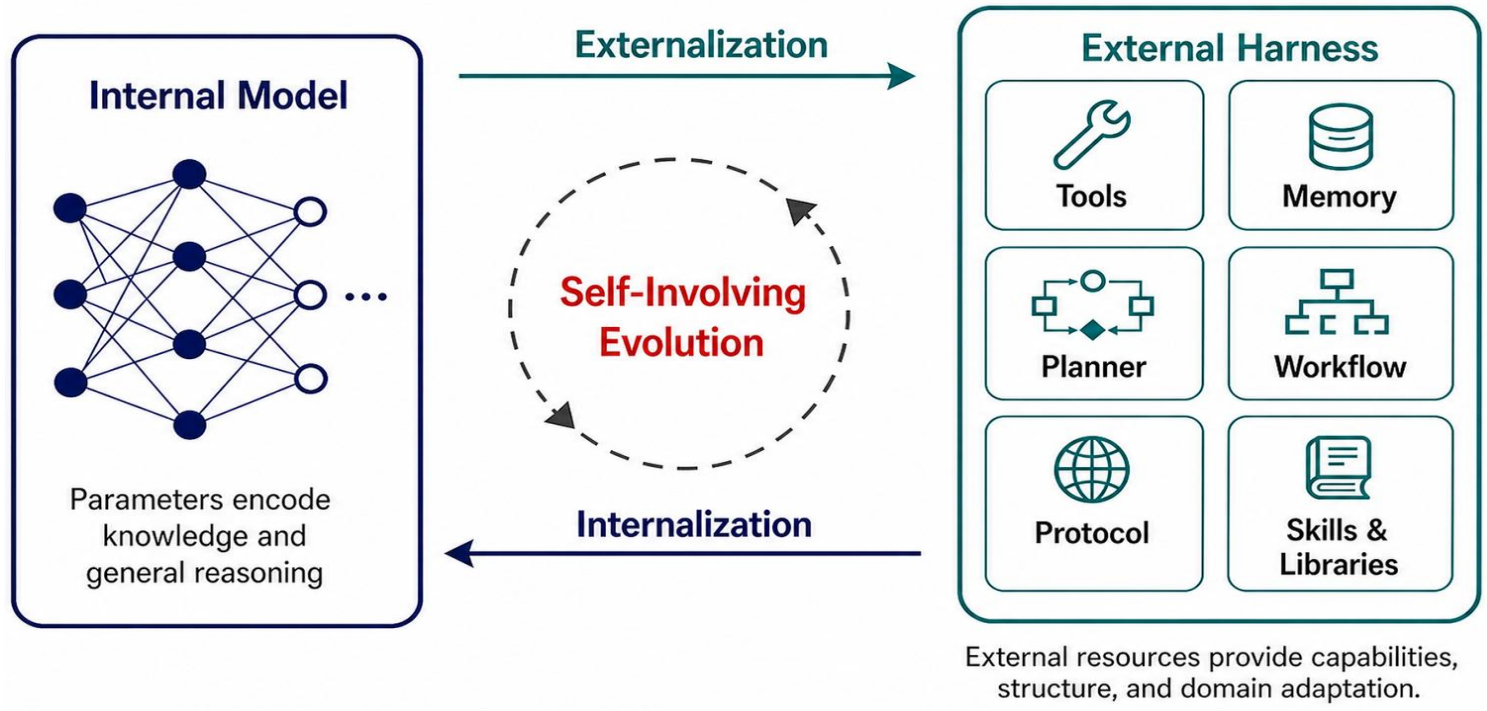
上海交通大学

智能计算研究院 博士生

chenyuzhou@sjtu.edu.cn

观点: 智能体进化需要全新的耦合视角

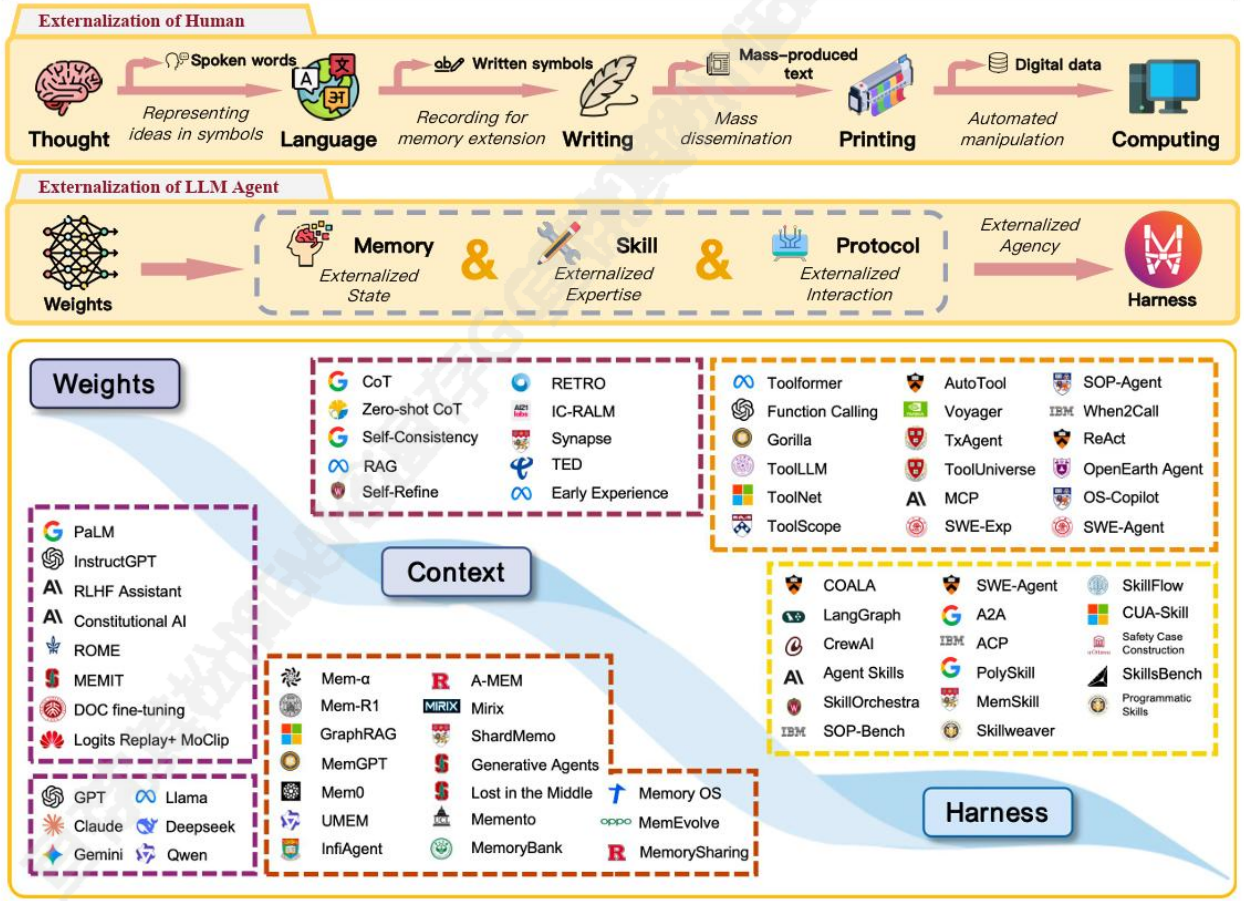
- **参数规模化扩张可以提升模型基础能力，却推高了场景适配的成本。**
  - 智能体在真实环境中的表现，仍依赖面向特定场景的工具、工作流与上下文信息。
- **外部编排框架承载着工具、工作流、记忆模块、API、通信协议与技能库。**
  - 外部资源是实现快速场景适配的核心支撑
- **有价值的交互轨迹可通过知识蒸馏反向注入模型内部。**
  - 经过验证的成功模式，将逐步内化为模型的原生能力。



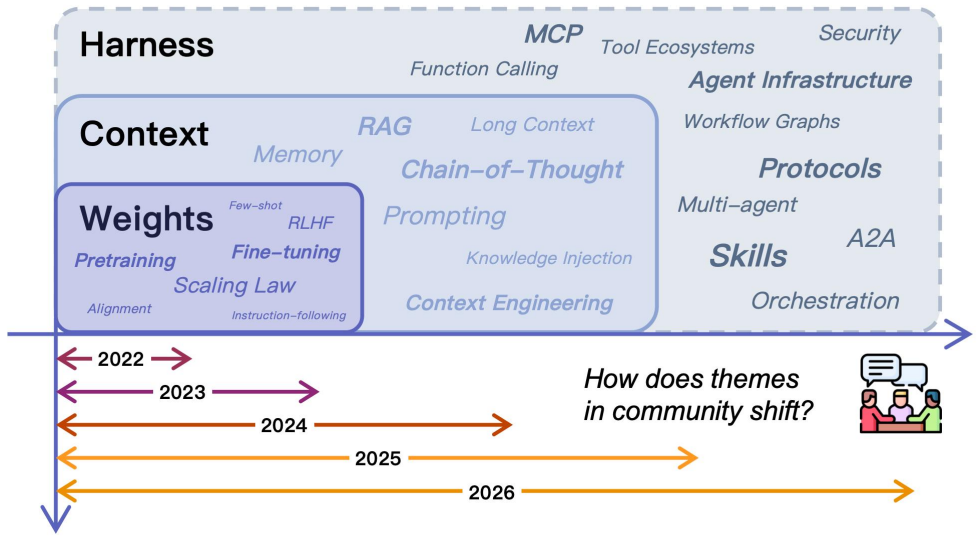
智能体的发展需要 model-harness 协同演化

Externalization：为什么智能体需要harness?

> *The power of the cognitive artifact comes from its function as a representation. ... Cognitive artifacts do not change human capabilities. They change the task.*
>
> *— Donald A. Norman*

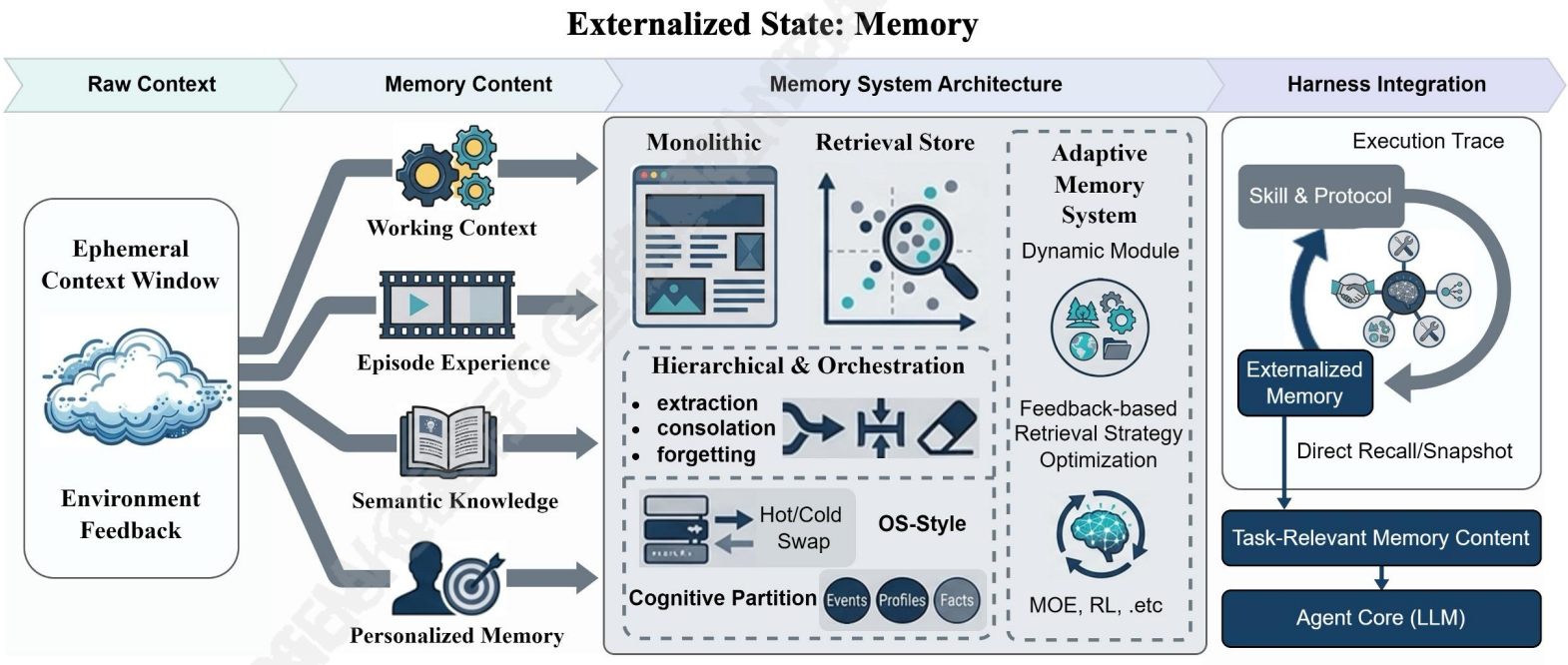


- **Memory**：把跨时间状态放到外部。
- **Skill**：把任务流程、启发式、约束写成可复用工件
- **Protocols**：把工具、用户、其他 agent 的交互变成结构化条约
- **Harness**：负责调度、权限、观察、控制和反馈。



Zhou, Chenyu, et al. "Externalization in llm agents: A unified review of memory, skills, protocols and harness engineering." *arXiv preprint arXiv:2604.08224* (2026).

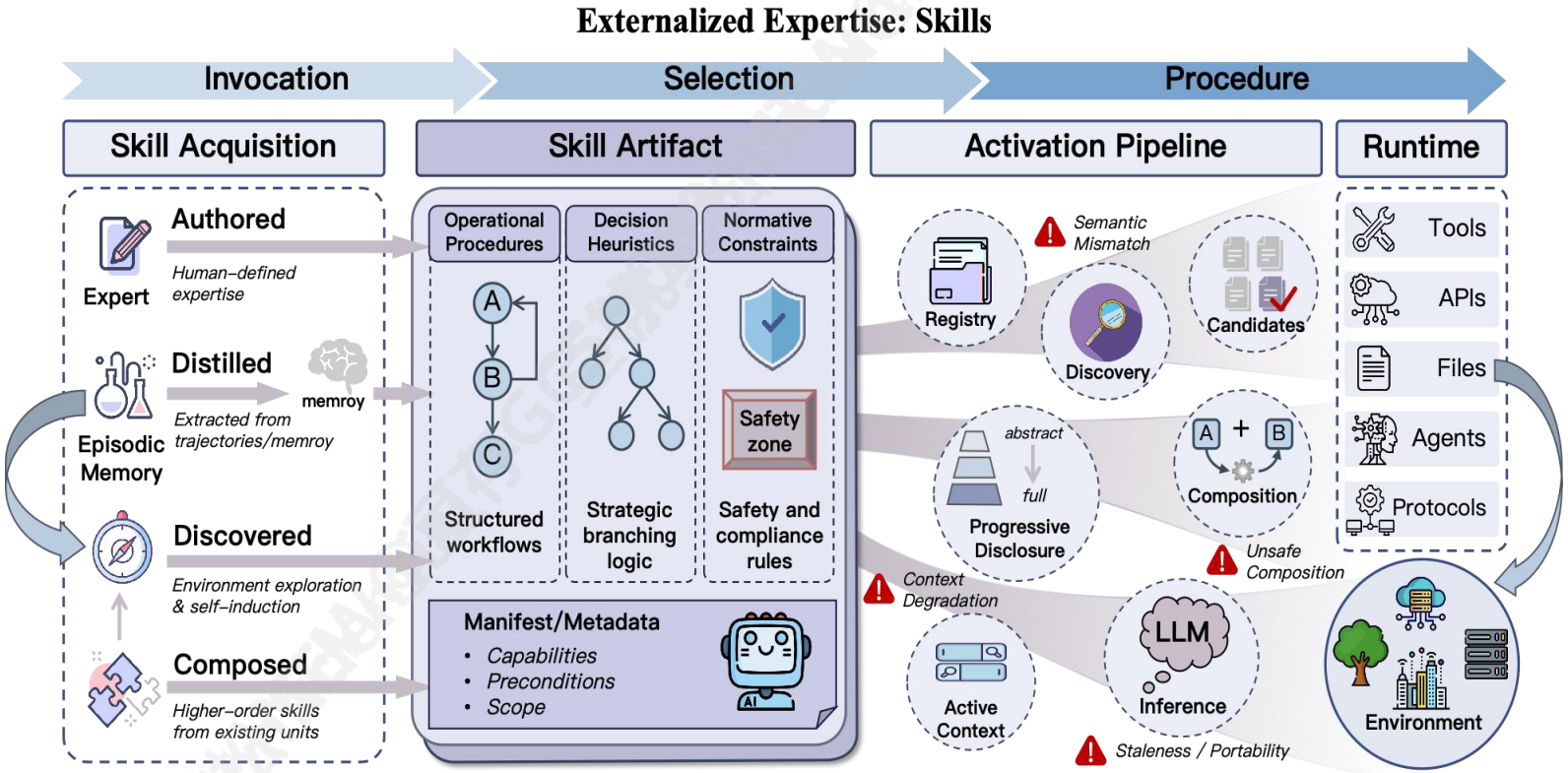
Memory：外化的状态



- **外化的负担**：Memory 外化解决 agency 的**时间负担**。
- **四层外化状态**：
  - Working Context (工作上下文): 当前任务的活跃中间
  - Episodic Experience (情景经验): 记录先前运行中发生了什么
  - Semantic Knowledge (语义知识): 超越任何单个 episode 的抽象
  - Personalized Memory (个性化记忆): 关于特定用户/团队/环境的稳定信息
- **关系**：
  - 重复出现的程序性规律可能先作为 episodic trace
  - 一旦 harness 将其提升为显式可复用指导，它们就不再是 memory 而属于 skill 层

Zhou, Chenyu, et al. "Externalization in llm agents: A unified review of memory, skills, protocols and harness engineering." *arXiv preprint arXiv:2604.08224* (2026).

Skill：外化的知识

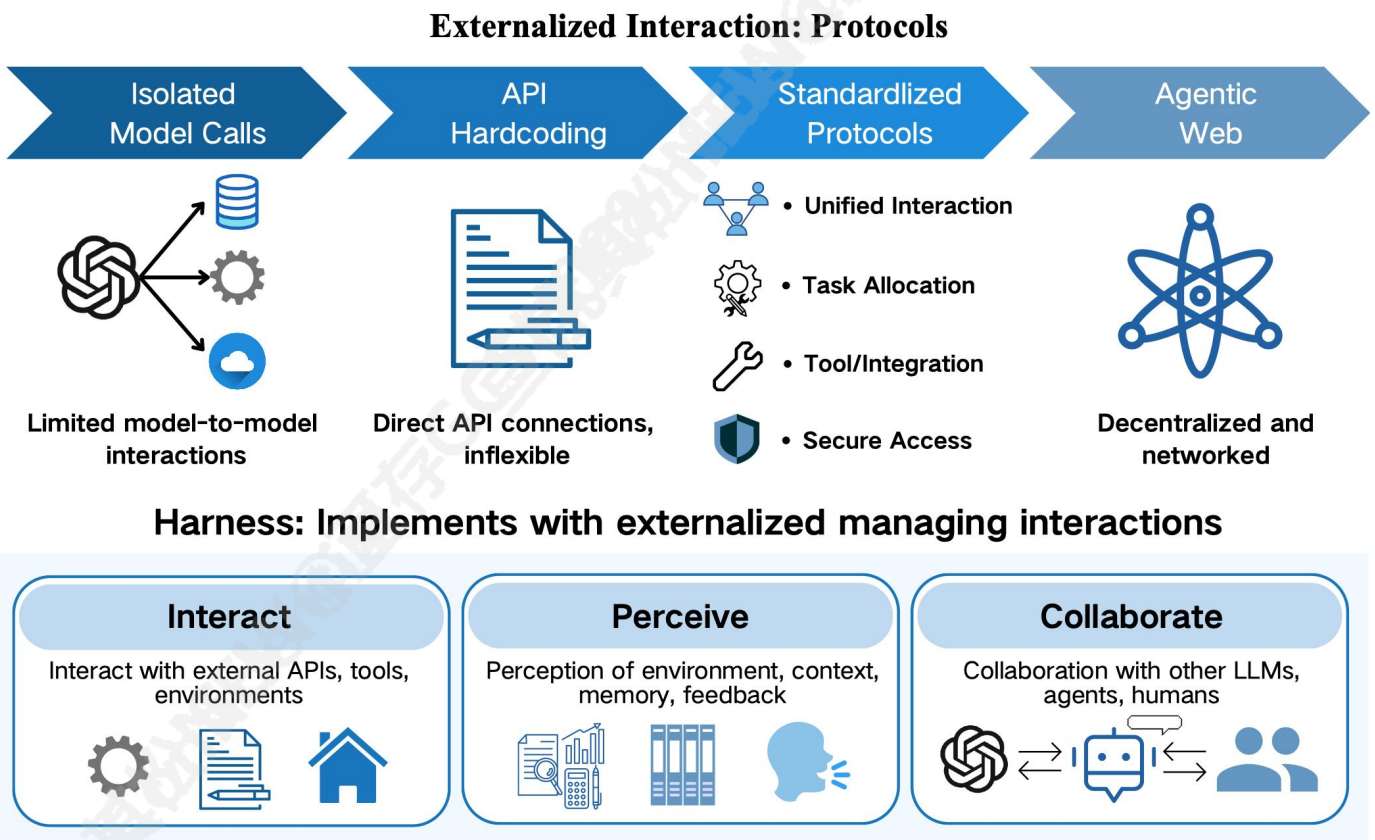


- **外化的负担**：从每次临场重构工作流，转向复用已有 procedure。
- **外化的内容**：
  - Operational procedures：任务步骤、依赖、停止条件
  - Decision heuristics：分支选择、失败恢复、优先级
  - Normative constraints：安全、合规、作用范围与边界
- **运行时作用**：
  - memory 提供情境证据
  - tools/protocols 提供执行接口
  - skill 组织任务流程

Skill是程序性知识外化的载体，运行时是否一定以文本形式注入context?

Zhou, Chenyu, et al. "Externalization in llm agents: A unified review of memory, skills, protocols and harness engineering." *arXiv preprint arXiv:2604.08224* (2026).

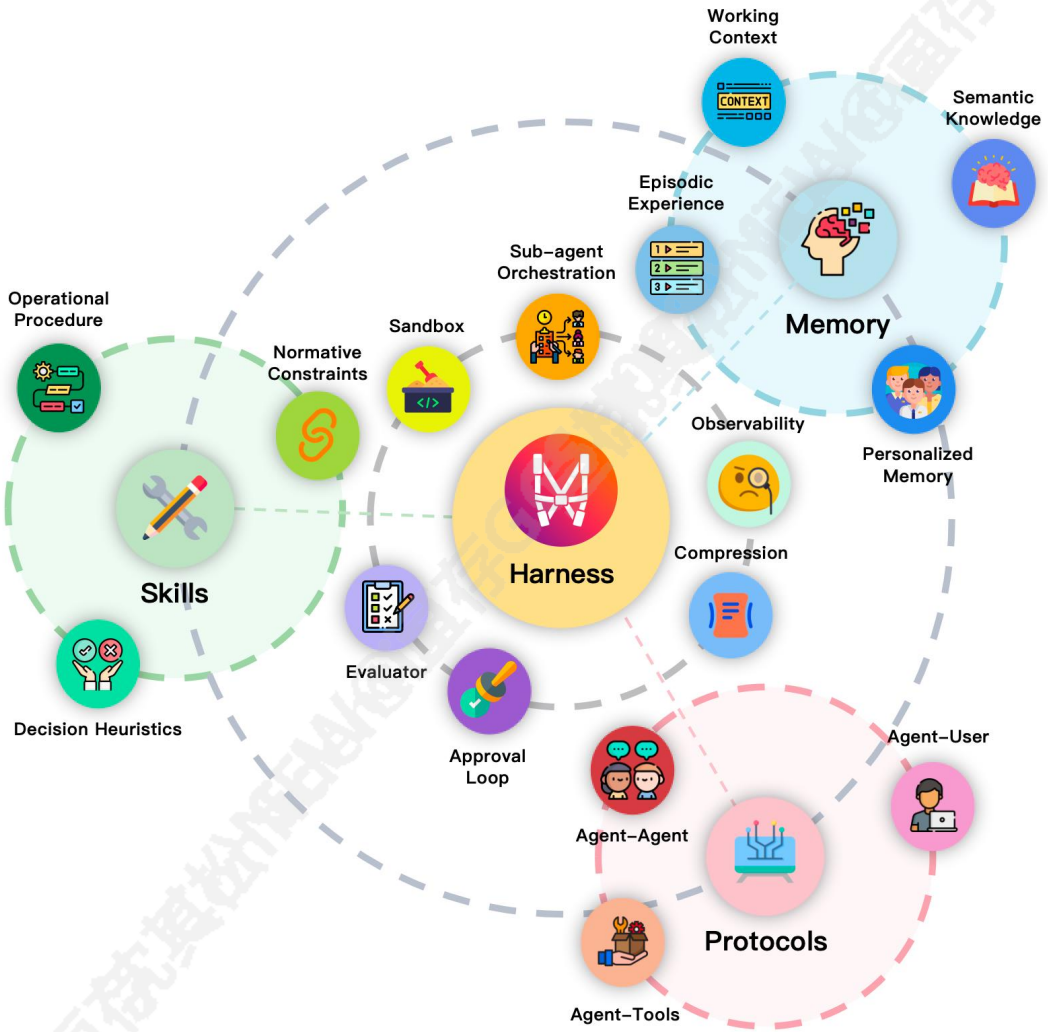
Protocols： 外化的交互

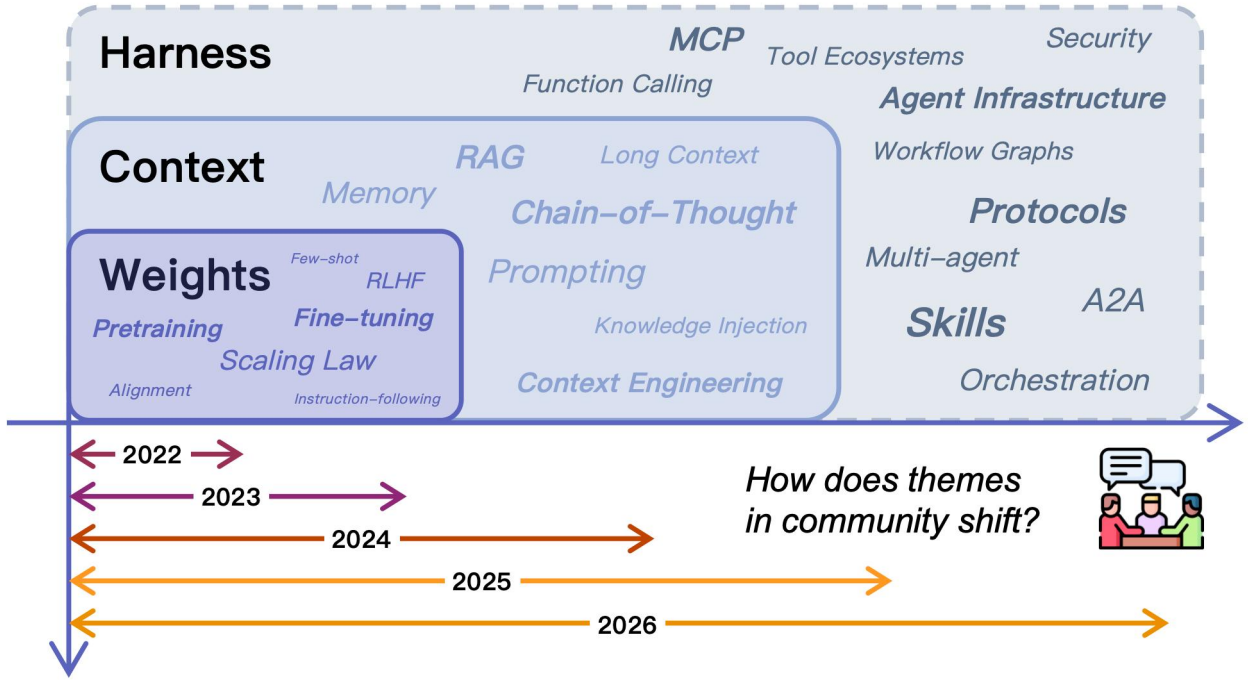


- **外化的负担**：从孤立模型调用，到 **Agentic Web**。
- **Harness 通过三个功能面实现外化交互管理:**
  - Interact: 与外部 API/工具/环境对接
  - Perceive: 感知环境/上下文/memory/反馈
  - Collaborate: 与其他 LLM/agent/人类协作
- **运行时作用**：
  - Memory 外化学到了什么;
  - Skills 外化如何执行;
  - Protocols 外化 memory 和 skills 进入世界所通过的**方式**

Zhou, Chenyu, et al. "Externalization in llm agents: A unified review of memory, skills, protocols and harness engineering." *arXiv preprint arXiv:2604.08224* (2026).

Harness：智能体的认知环境





Agent = Model + Harness

Zhou, Chenyu, et al. "Externalization in llm agents: A unified review of memory, skills, protocols and harness engineering." *arXiv preprint arXiv:2604.08224* (2026).

从 Externalized Skills 到 Latent Skills

从潜空间视角理解、控制与组合智能体技能

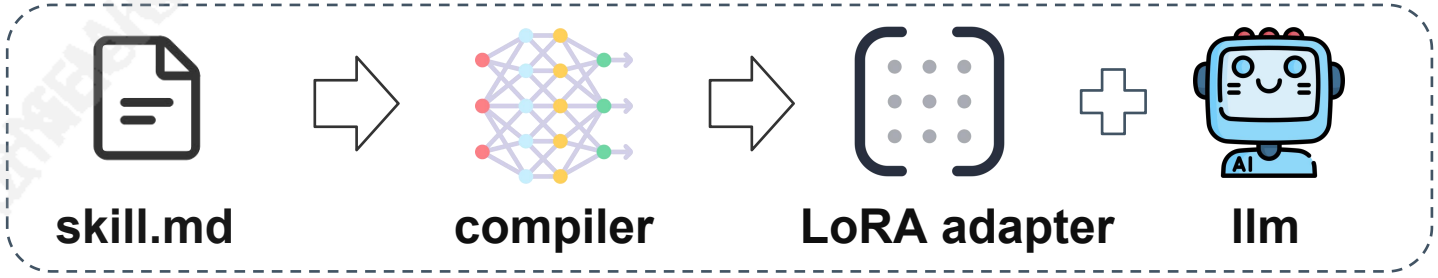
从 Textual Skill 到 Latent Skill

- **Textual Skill**

⚠ 重复开销
⚠ 明文暴露
⚠ 粗粒度使用

将Skill作为外部文本注入上下文，通过prompt影响模型行为。

- **Latent Skill**



✓ 低运行成本
✓ 低安全风险
✓ 细粒度使用

将Skill作为潜在权重挂载到模型，通过parameter影响模型行为

方法定义：从文本条件到参数条件

LatentSkill 的核心：不再让模型每一步读 skill text，而是用 compiler 把 skill 编译成 LoRA，通过参数影响模型行为。定义为 skill compiler Gϕ，将 skill 文档映射为 LoRA updates

- **Compiler：**

$$\Delta_s = G_\phi(s)$$

- 从文本条件到参数条件：

$$p_{\theta,\phi}(y_t \mid h_t, s) = p_{\theta \oplus \alpha\Delta_s}(y_t \mid h_t)$$

- **LoRA 更新形式：**

$$\Delta W_s^{(m)} = B_s^{(m)} A_s^{(m)}$$

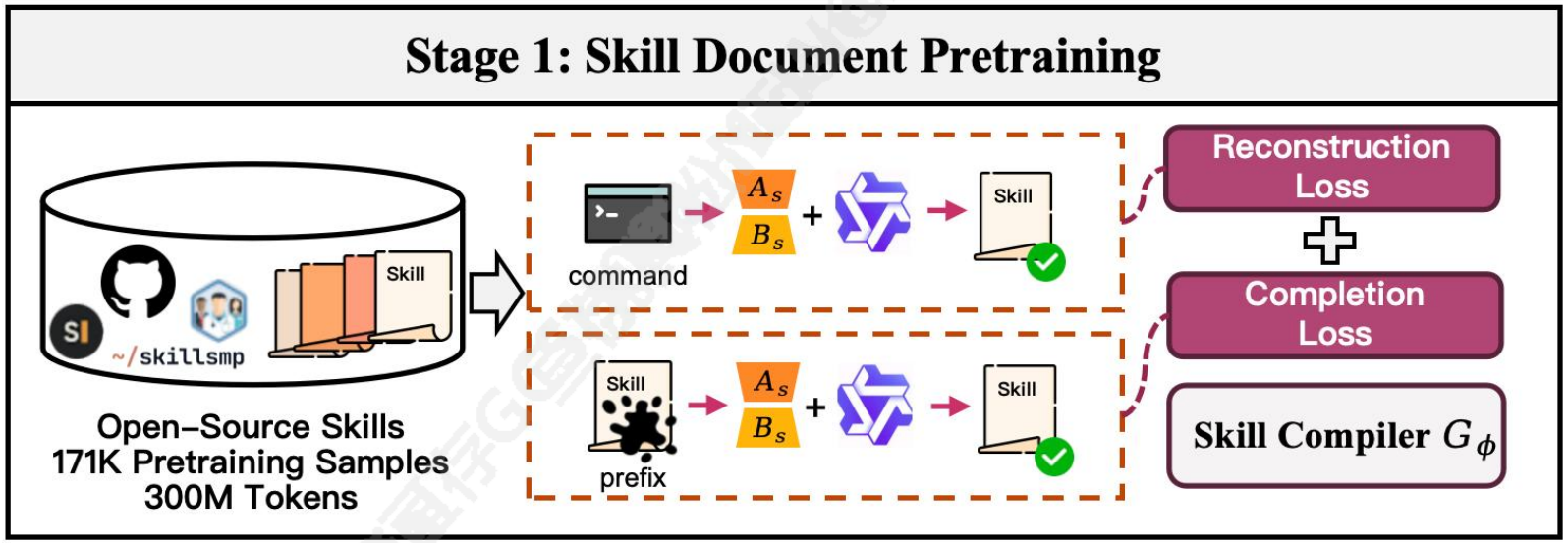
- 挂载 **frozen llm：**

$$W' = W + \frac{\alpha}{r} B_s A_s$$

- $\boldsymbol{s}$：textual skill document
- $\boldsymbol{G_\phi}$: skill compiler
- $\Delta_{\boldsymbol{s}}$: 生成的 latent skill / LoRA updates
- α: skill 注入强度

两阶段训练：LatentSkill 如何学会编译Skill?

Stage 1: 让 compiler 建立 text → adapter 的基础映射能力



$$\Delta_i^{\text{pre}} = G_\phi(s_i^{\text{src}})$$

$$\mathcal{L}_{\text{pre}} = -\sum_{i,j} \log p_{\theta \oplus \alpha \Delta_i^{\text{pre}}}(z_{i,j} \mid q_i, z_{i,<j})$$

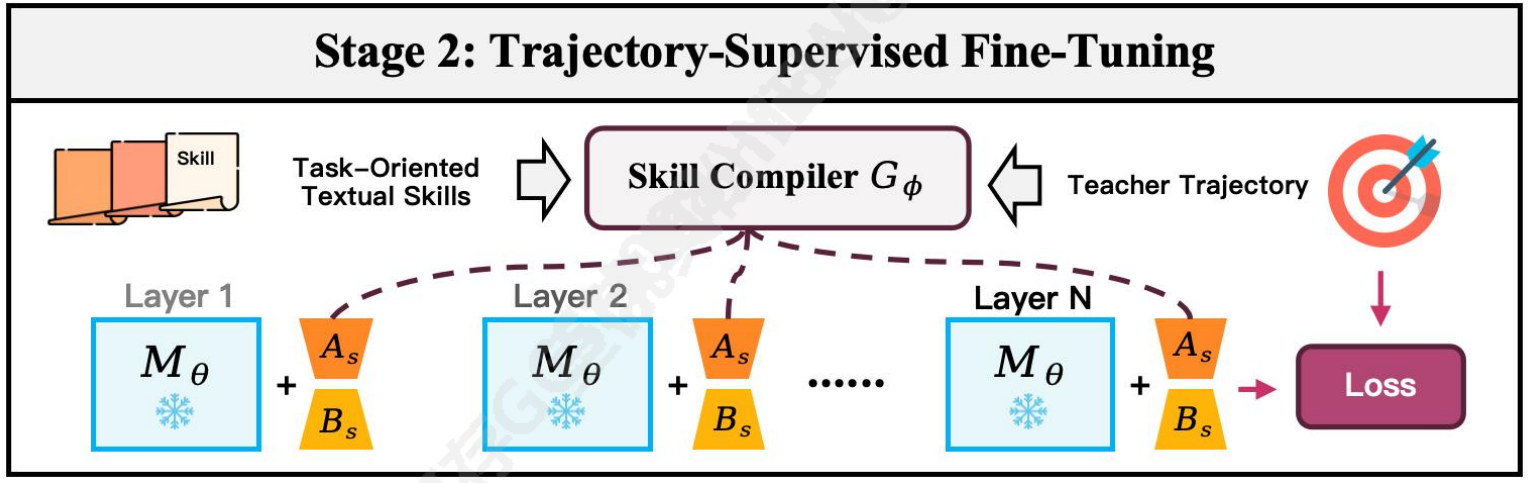
- s_i^{src}：完整 skill 或截断 prefix
- q_i：重构 / 补全指令
- z_i：目标 skill 文本
- 只更新 compiler G_ϕ，冻结 backbone

- 数据：大规模真实 Skill 文档
  - 17 万条开源 skill documents，来自 GitHub、SkillsMP 等，覆盖不同任务场景。
  - 直接用 skill 文本本身构造自监督学习信号。
- 任务：自监督 Skill 编译任务
  - 重构：完整 skill 文档 → adapter → 还原原始 skill 内容。
  - 补全：部分 skill 文档 → adapter → 补全完整 procedure。

先学会把程序性知识从文本写入权重，形成可承载skill的latent adapter

两阶段训练：LatentSkill 如何学会编译Skill?

Stage 2: 让 adapter 把静态 skill 表示对齐到真实 agent 行为



$$\tau_i = \{(h_{i,t}, y^*_{i,t})\}_{t=1}^{T_i}$$

$$\Delta_i^{\text{sft}} = G_\phi(s_i)$$

$$\mathcal{L}_{\text{sft}} = -\sum_{i,t,j} \log p_{\theta \oplus \alpha \Delta_i^{\text{sft}}} \left(y^*_{i,t,j} \mid h_{i,t}, y^*_{i,t,<j}\right)$$

- $h_{i,t}$: 第 t步 history / observation
- $y^*_{i,t}$: teacher output

- 数据：带 Skill 标注的 Teacher 轨迹
  - 以 OpenAI o3 作为teacher model，用于监督 LatentSkill 对齐 agent 行为。
  - 包含任务目标、历史观察、teacher reasoning/action 和环境反馈。
- 目标：从文本表示到行为对齐
  - 在每个交互步骤，根据当前 history/observation 模仿 teacher 的思考与动作。
  - 同一个 adapter 贯穿多步交互，让 LatentSkill 学到稳定的 skill-level policy。

再从承载程序性知识的 adapter，变成能够驱动 agent 行为的潜在能力模块

实验设置

- **Model & Skill Carrier**：
  - Backbone：冻结 **Qwen3-8B**。
  - Key baseline：**In-context Skill**，使用同一份 skill，但放在 prompt 中。
  - Ours：同一份 skill 经 compiler 转成 LoRA adapter，挂载在llm上。
- **Skill Matching**：
  - ALFWorld：按任务类别匹配 skill，Pick 和 Pick2 共享 Pick skill。
  - Search-QA：3 类 skill，对应 direct retrieval / multi-hop / comparison。
- **Benchmarks**：
  - ALFWorld：6 类家居多步交互任务，评估 success rate / steps / token cost。
  - Search-QA：7 个搜索问答数据集，覆盖 single-hop 和 multi-hop，评估 EM / token cost
- **Evaluation**：
  - ALFWorld：seen 140个episode、unseen 134个episode，每个 episode 最多 50 步。
  - Search-QA：每数据集 500 条，Bamboogle 125 条；E5 top-3 检索，最多 4 次 retrieval。

实验结果：能力收益保留，运行成本下降

Table 1: Performance on **ALFWorld** in success rate %. Results are reported on both *seen* and *unseen* splits with per-task breakdown. *Step* denotes the average number of steps taken per episode. *Prefill* and *Decode* report average token counts (k) per step. The best and second-best results are highlighted in **bold** and underline, respectively.

Method	ALFWorld Task						Avg↑	Step↓	Cost↓	
	Pick	Look	Clean	Heat	Cool	Pick2			Prefill	Decode
Seen split										
Vanilla	82.9	46.2	18.5	37.5	32.0	29.2	43.6	35.0	**0.44**	0.55
Few-Shot	82.9	46.2	44.4	**68.8**	36.0	12.5	50.0	31.3	2.04	0.54
Reflexion	77.1	53.9	33.3	31.3	56.0	12.5	46.4	33.3	0.55	0.66
AdaPlanner	91.4	30.8	33.3	37.5	28.0	33.3	47.1	35.9	0.54	0.84
In-Context Skill	85.7	69.2	**70.4**	31.3	12.0	33.3	52.9	30.8	1.21	0.50
LatentSkill	**97.1**	**92.3**	63.0	43.8	**64.0**	**75.0**	**74.3**	**28.4**	**0.44**	**0.34**
Unseen split										
Vanilla	54.2	55.6	41.9	47.8	57.1	23.5	47.0	34.9	**0.44**	0.62
Few-Shot	66.7	50.0	61.3	**56.5**	76.2	0.00	54.5	**28.5**	2.04	**0.50**
Reflexion	50.0	61.1	58.1	43.5	66.7	0.00	48.5	32.8	0.57	0.82
AdaPlanner	83.3	50.0	54.8	39.1	52.4	29.4	53.0	34.3	0.54	0.75
In-Context Skill	70.8	61.1	**74.2**	43.5	47.6	23.5	56.0	29.7	1.23	0.61
LatentSkill	**91.7**	**66.7**	64.5	43.5	**81.0**	**70.6**	**69.4**	31.4	**0.44**	0.51

- 成功率提升
  - Seen Avg 52.9 → **74.3**
  - Unseen Avg 56.0 → **69.4**
- 上下文成本下降
  - Prefill Avg 1.2k → **0.44k**
  - skill 不再每一步重复进入 context
- 执行过程更高效
  - 平均路径更短，决策更高效
  - 改变的是多步决策过程，不只是最终结果

在ALFWorld长程交互任务中，LatentSkill 不只是减少 prompt token，而是让 agent 更稳定地完成动作序列

实验结果：能力收益保留，运行成本下降

Table 2: Performance on **Search-QA** in exact match %. † and ⋆ indicate in-domain and out-of-domain datasets, respectively. *Cost* denotes average token counts (k) per step. The best and second-best results are highlighted in **bold** and underline, respectively.

Method	Single-Hop QA			Multi-Hop QA				Avg↑	Cost↓
	NQ†	Triv⋆	Pop⋆	Hotp†	2WK⋆	MuS⋆	Bam⋆		
Vanilla	25.2	50.6	35.2	26.2	26.8	4.20	28.8	28.1	0.24
CoT	19.2	50.8	20.0	22.4	26.2	5.00	37.6	24.5	**0.09**
Few-Shot	34.6	57.6	39.4	30.0	25.8	6.00	17.6	31.7	0.94
R1-Instruct	27.0	55.0	31.6	26.6	33.6	5.80	34.4	30.1	0.24
RAG	**39.0**	**64.0**	**45.0**	32.4	21.2	6.80	27.2	34.4	0.89
In-Context Skill	27.2	56.4	33.0	30.2	**39.8**	7.60	**38.4**	32.6	1.10
`LatentSkill`	36.2	57.6	41.0	**39.6**	32.0	**9.80**	25.6	**35.6**	0.31

- 整体效果提升
  - EM Avg 32.6 → **35.6**
- 运行成本更低
  - Cost Avg 1.10k → **0.31k**
- 搜索策略迁移
  - 保留分解问题、构造查询、整合证据等策略性行为。

在Search-QA搜索问答任务中，LatentSkill 仍能保留分解问题、构造查询、整合证据等策略性行为

Structured：LatentSkill 压缩低秩有效

Table 8: Frobenius norm ($\|\Delta W\| \times 10^{-3}$) and stable rank (SR) of hypernetwork-generated ΔW at Pretrain and SFT stages. SFT columns are shaded in blue.

	$\|\Delta W\|$ ($\times 10^{-3}$)		Stable Rank	
Skill	**Pretrain**	**SFT**	**Pretrain**	**SFT**
ALFWorld				
Clean	2.788	2.843	2.36	2.18
Cool	2.788	2.841	2.36	2.18
Heat	2.785	2.841	2.35	2.17
Look	2.787	2.841	2.36	2.18
Pick	2.787	2.841	2.35	2.17
Search				
Direct	2.790	2.848	2.40	2.23
MultiHop	2.793	2.849	2.39	2.23
Compare	2.790	2.846	2.40	2.23

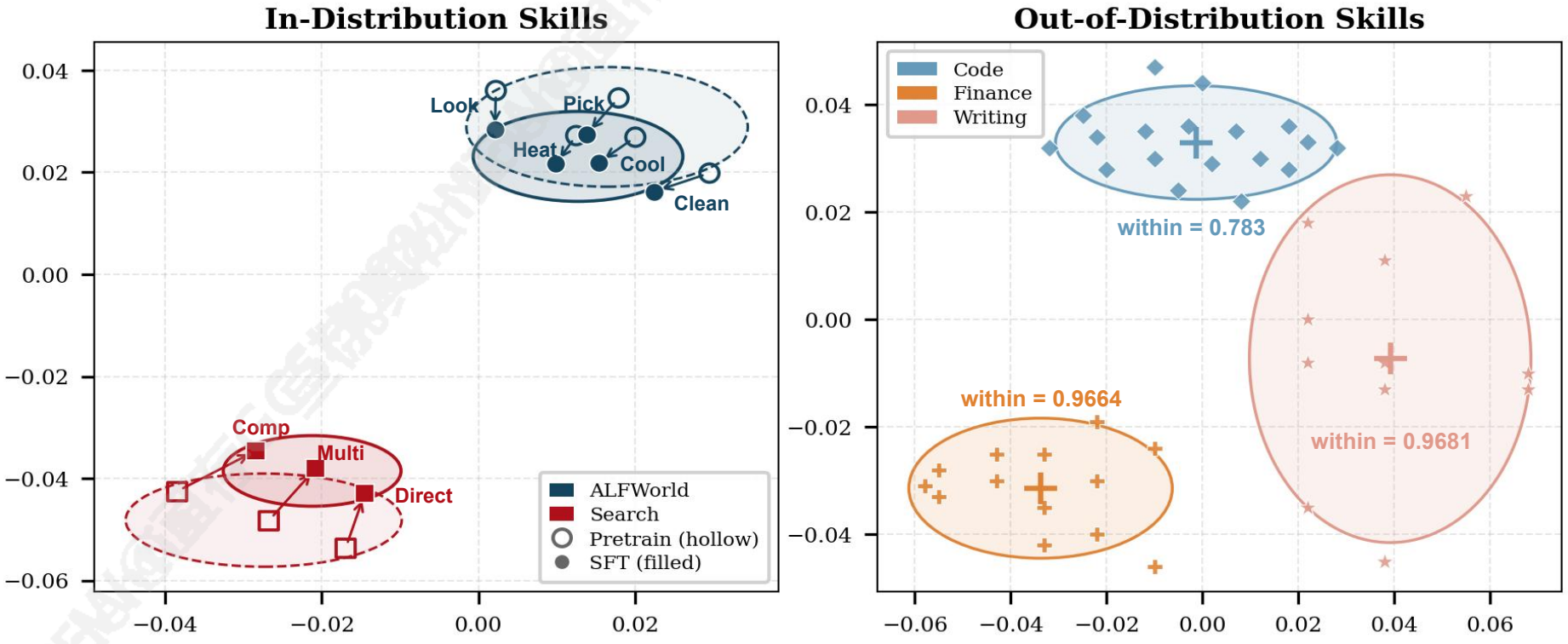
Table 9: Cumulative singular value energy ratio (%) of top-k directions at Pretrain and SFT stages. SFT columns are shaded in blue.

Skill	**Rank-1** (%)		**Rank-2** (%)		**Rank-5** (%)	
	Pre.	**SFT**	**Pre.**	**SFT**	**Pre.**	**SFT**
ALFWorld						
Clean	48.8	54.1	67.1	70.6	92.9	93.7
Cool	48.8	54.2	67.0	70.7	92.9	93.7
Heat	49.1	54.3	67.3	70.8	93.0	93.8
Look	48.9	54.2	67.1	70.7	92.8	93.7
Pick	49.2	54.4	67.4	70.9	93.0	93.8
Search						
Direct	48.0	53.2	66.4	69.7	92.5	93.1
MultiHop	48.2	53.1	66.5	69.7	92.6	93.1
Compare	48.1	53.2	66.4	69.7	92.6	93.1

- **有效秩低：**
  - 不同skill的ΔW幅度接近，说明compiler输出稳定。
  - Stable Rank仅约2.2，SFT后进一步降低，说明 skill update 只占少数有效方向。
- **能量高度集中：**
  - Top-2约67–71%，Top-5约93%，承载主要信号。
  - SFT后进一步上升，说明行为对齐把skill信号压缩到更少、更集中的方向。

LatentSkill是把程序性知识压缩成低秩、紧凑、可分析的latent adapter

Structured： LatentSkill 形成语义几何



- **域内聚集**：ALFWorld/Search在权重空间中形成不同区域，LatentSkill捕获的不只是词面相似。
- **域外泛化**：Code/Finance/Writing等未见领域也能形成结构，Compiler学到程序性知识表示。

让skill不再只是prompt文本，而是可以被度量、比较和分析的潜在能力对象

Controllable： LatentSkill 强度可调

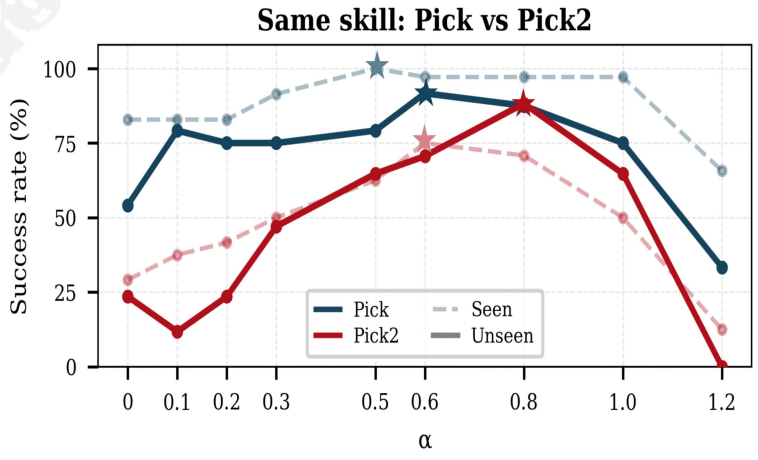
通过 LoRA scaling coefficient α，LatentSkill 可以连续调节 skill 对模型行为的影响

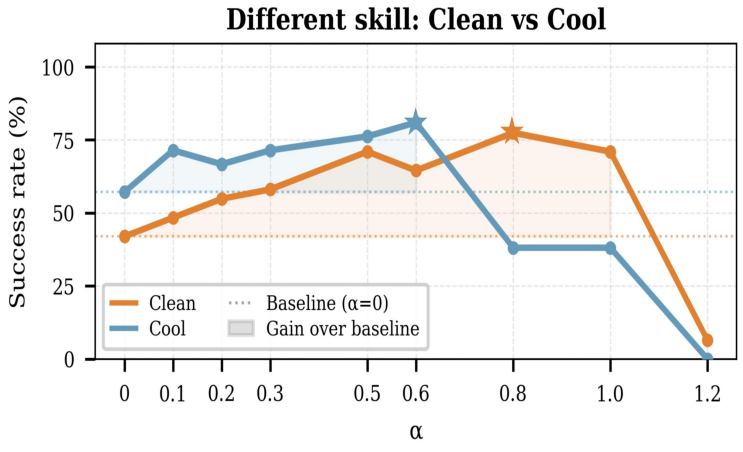
Seen

α	Pick	Look	Clean	Heat	Cool	Pick2	Avg
0.0	82.86	46.15	18.52	37.50	32.00	29.17	43.57
0.1	82.86	53.85	33.33	43.75	52.00	37.50	52.86
0.2	82.86	61.54	44.44	50.00	48.00	41.67	56.43
0.3	91.43	69.23	51.85	**56.25**	48.00	50.00	62.86
0.5	**100.0**	**92.31**	48.15	43.75	56.00	62.50	68.57
0.6	97.14	**92.31**	62.96	43.75	**64.00**	**75.00**	**74.29**
0.8	97.14	76.92	**70.37**	31.25	52.00	70.83	70.00
1.0	97.14	61.54	66.67	37.50	44.00	50.00	63.57
1.2	65.71	7.69	7.41	6.25	8.00	12.50	22.86

Unseen

α	Pick	Look	Clean	Heat	Cool	Pick2	Avg
0.0	54.17	55.56	41.94	47.83	57.14	23.53	47.01
0.1	79.17	55.56	48.39	43.48	71.43	11.76	52.99
0.2	75.00	55.56	54.84	47.83	66.67	23.53	55.22
0.3	75.00	55.56	58.06	**65.22**	71.43	47.06	62.69
0.5	79.17	**77.78**	70.97	56.52	76.19	64.71	**70.90**
0.6	**91.67**	66.67	64.52	43.48	**80.95**	70.59	69.40
0.8	87.50	66.67	**77.42**	34.78	38.10	**88.24**	65.67
1.0	75.00	72.22	70.97	43.48	38.10	64.71	61.19
1.2	33.33	0.00	6.45	4.35	0.00	0.00	8.21





- **性能随 α 呈 “倒U型”曲线**：存在有效注入区间，适中 scaling 有效，过强 scaling 会破坏原模型能力。
- **不同任务的最佳 α 不同，弱 baseline 往往需要更强 skill**：harness 不只要选择哪个 skill，还要选择 skill 介入模型的强度。

把 skill use 从 prompt 级别的是否加载，变成 weight 级别的强度控制

Composable：有效组合依赖语义组件对齐

实验设置：以 Look 为目标 skill，引入 Pick 的互补搜索能力，比较 Look-Only、Pick-Only、Direct Merging、Text Merging 和 Component Merging。

Table 3: Skill composition results on all 31 Look task episodes under five skill composition configurations. The best result per split is highlighted in **bold**.

	Seen		Unseen	
Method	**Ep.**	**%**	**Ep.**	**%**
Look-Only	8/13	61.5	13/18	72.2
Pick-Only	8/13	61.5	11/18	61.1
Direct Merging	9/13	69.2	11/18	61.1
Text Merging	8/13	61.5	11/18	61.1
Component Merging	11/13	**84.6**	14/18	**77.8**

- 朴素组合会干扰：
  - 直接合并完整 LoRA 或直接拼接 skill text会带来行为冲突。
  - Direct / Text Merging 在 Unseen 上均为 61.1，低于 Look-Only 的 72.2。
- 组件组合更有效：
  - Component Merging 达到 84.6 / 77.8，相比 Look-Only 提升 +23.1 / +5.6 pp。
  - 对齐后的组件组合能引入互补能力，同时减少干扰。

LatentSkill 可以在 LoRA 空间中组合多个技能，但有效组合依赖语义组件对齐，而不是直接相加

Robust & Secure：Prompt 外运行，更稳定、更低暴露

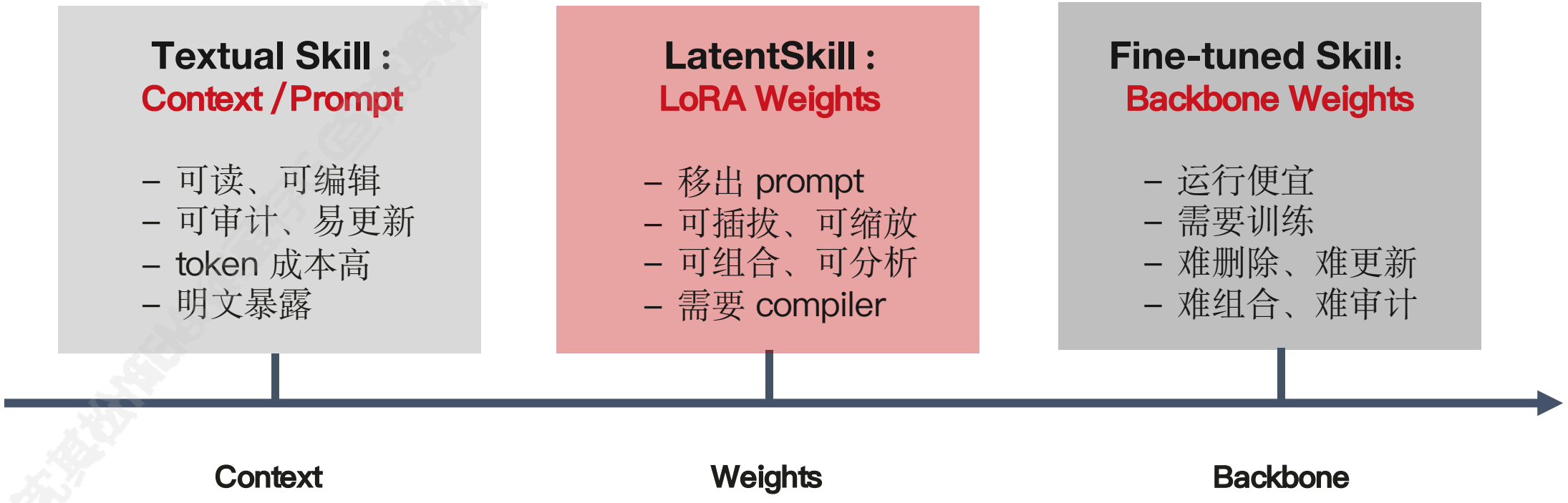
Table 4: Average performance under skill text perturbations and adversarial attacks. ALFWorld reports success rate (%) and Search-QA reports exact match (%).

	ALFWorld		Search-QA	
Method	**In-context**	**Latent**	**In-context**	**Latent**
Base	52.9	**74.3**	32.6	**35.6**
Paraphrase	50.7	**67.9**	33.2	**34.0**
Plaintext	50.0	**74.3**	32.4	**34.4**
Reorder	50.7	**69.3**	32.4	**33.6**
Noise	47.9	**71.4**	31.7	**33.7**
Hijack	8.57	**38.6**	23.5	**34.0**
Extract	48.6	**70.0**	21.3	**29.3**

- **文本扰动下更稳定：**
  - Paraphrase / Plaintext / Reorder / Noise 主要改变 skill 的措辞、格式、顺序和冗余信息。
  - LatentSkill 依赖编译后的 LoRA 表示，较少受文本表面形式影响。
- **Prompt 攻击下更安全：**
  - Hijack / Extract 主要攻击 prompt 中可见的明文 skill。
  - LatentSkill 不直接暴露 skill 文本，降低被覆盖、被抽取的风险，保护 skill 版权。

LatentSkill 在保留语义作用的同时降低了文本扰动和明文暴露风险

LatentSkill：外化与内化之间



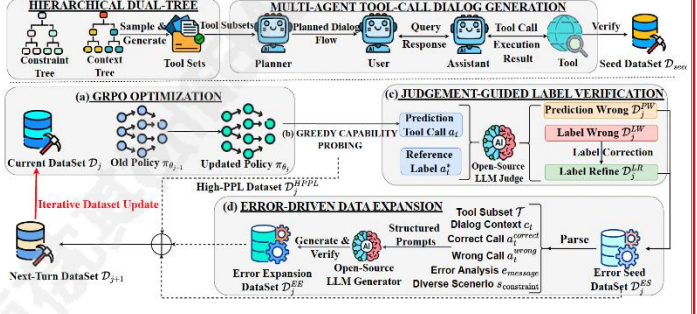
LatentSkill 对 Agent 平台意味着什么？

- **更低成本**
  - 长程任务中不再每步重复读取 skill text，减少隐性的 prefill 开销。
- **更低暴露**
  - Skill 不再以明文出现在 prompt / 请求日志中，降低策略被直接抽取和复制的风险。
- **更好治理**
  - Skill 变成可插拔、可缩放、可组合的 adapter，便于版本管理、灰度发布和 A/B 测试。
- **新的产品形态**
  - 过去交付 markdown 手册，分发几乎等于暴露；现在可以交付 LoRA adapter，让 skill 更接近可授权、可计费、可撤回的软件模块。

路线图: Model-Harness 协同演化

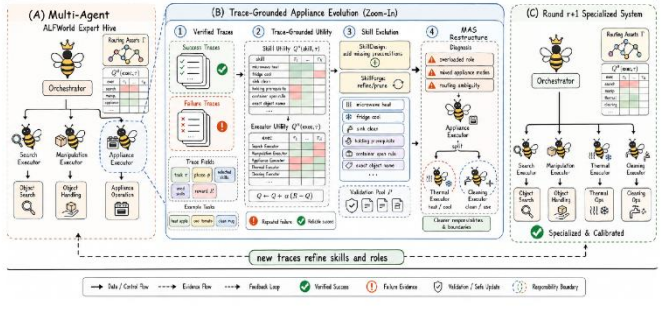
Model自演化

- **能力自感知:** 模型在执行任务前，应持续评估自身当前的优势、局限和置信水平。
- **自生成改进数据:** 模型能够自主构造训练数据和评测数据，用于后续能力提升。
- **自训练与自我优化闭环:** 模型利用生成数据、外部反馈和性能结果，持续微调或更新自身行为。



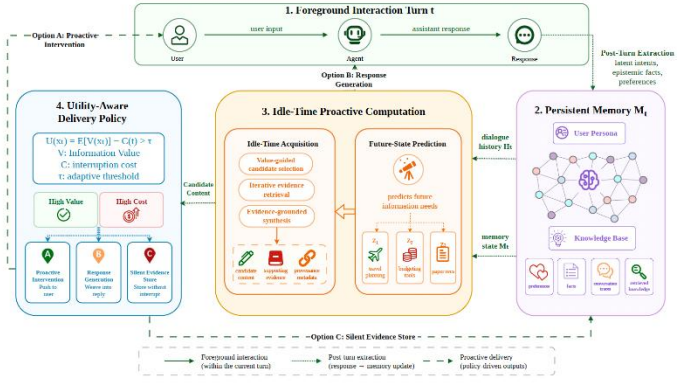
Harness自演化

- **人工工程难以规模化:** 每进入一个新领域，都需要专家从头设计 prompt、工具 schema 和工作流。
- **静态脚手架会退化:** 针对某个模型版本、任务分布或环境调优的 harness，一旦这些条件变化，就可能悄然失效。
- **模型能力演进快于外部封装:** 前沿模型能力不断提升，但围绕模型构建的工具、流程和框架往往难以及时跟上。

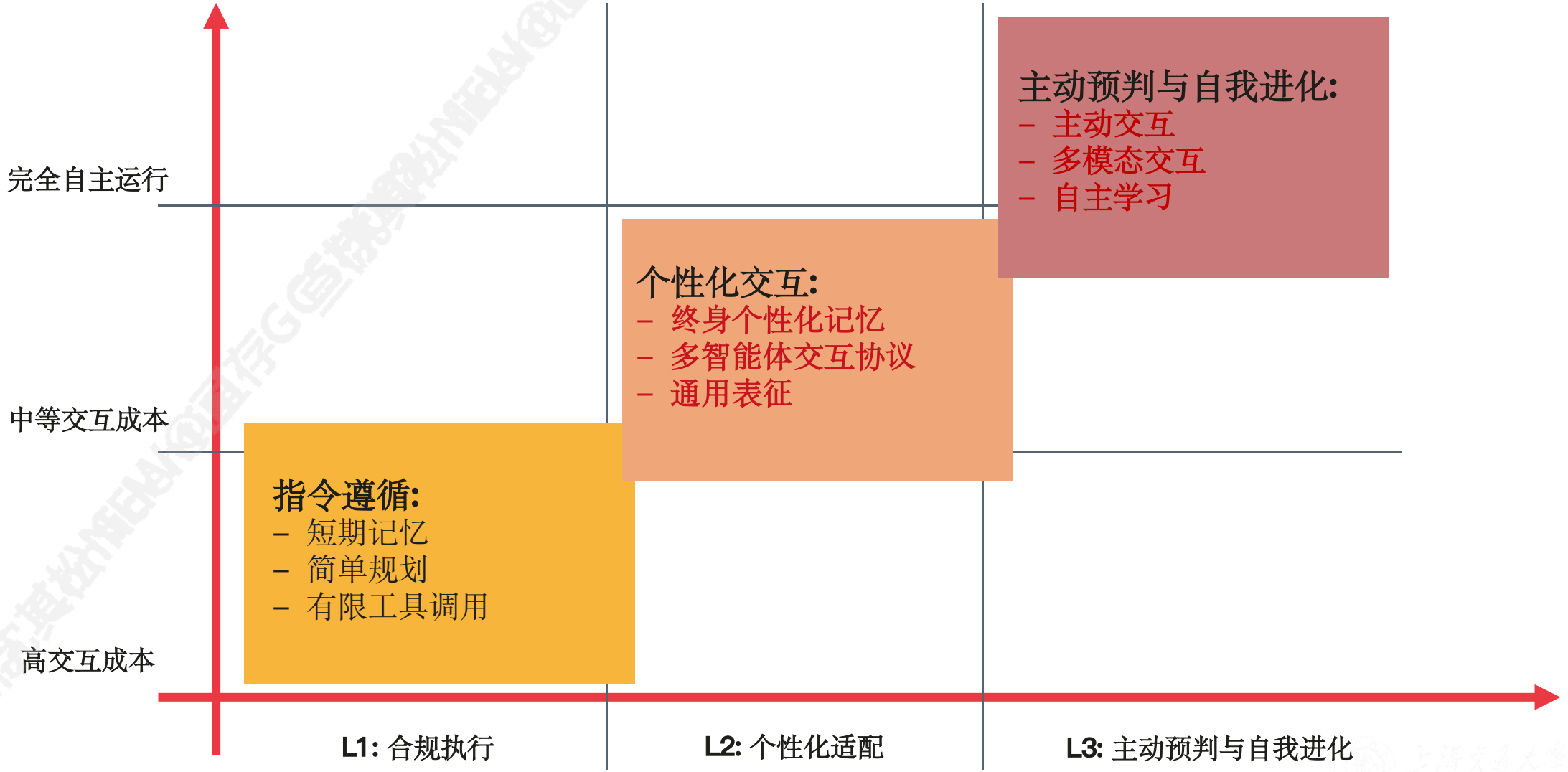


主动交互

- **主动交互范式:** 从被动接收用户指令，转向主动预测用户任务与需求。
- **新的交互范式:** 将上下文工程的负担从用户转移给 agent，显著降低用户的交互成本。
- **主动推断用户目标:** agent 可以预测用户需求，自动补全必要信息，并高效完成任务。



未来方向


完全自主运行
中等交互成本
高交互成本
主动预判与自我进化:
– 主动交互
– 多模态交互
– 自主学习
个性化交互:
– 终身个性化记忆
– 多智能体交互协议
– 通用表征
指令遵循:
– 短期记忆
– 简单规划
– 有限工具调用
L1: 合规执行
L2: 个性化适配
L3: 主动预判与自我进化

Thanks!

Email: chenyuzhou@sjtu.edu.cn

Homepage: https://0xzhouchenyu.github.io