SKILLALIGN: Aligning Skill Interfaces for LLM-based Agents

Shuo Ren[1], Xiaomian Kang[1], Jiajun Zhang[1,2,3]*
[1]Institute of Automation, Chinese Academy of Sciences
[2]School of Artificial Intelligence, University of Chinese Academy of Sciences
[3]Wuhan AI Research
shuo.ren@ia.ac.cn, {xiaomian.kang,jjzhang}@nlpr.ia.ac.cn

Abstract

Language-model agents increasingly rely on skills: reusable procedural knowledge for reasoning, tool use, and interaction. Existing work studies how skills are acquired, retrieved, compressed, or composed, but often assumes that once a skill is selected, its interface to the agent is fixed. We argue that this overlooks a key source of skill utility: the same skill can help, distract, or mislead depending on how it is exposed. We propose SKILLALIGN, a provider-agnostic framework that represents candidate skills as multi-view procedural cards and renders them through alternative exposure interfaces, including full instructions, hints, compressed summaries, workflows, or no exposure. This enables counterfactual evaluation where the task, agent, and candidate skills are fixed while only the exposure interface varies. Across ALFWorld and SkillsBench, we show that exposure form substantially affects task success and rendered context cost, and that compact top-k exposure can outperform full-library injection. We further conduct a replay-based policy-learning analysis on ALFWorld, showing that adaptive exposure contains learnable signal but remains far from oracle selection. Our results suggest that skill-augmented agents should optimize not only which skills to use, but also how those skills are presented.

1 Introduction

Large language model agents increasingly rely on **skills**: reusable procedural knowledge that can guide agents through recurring patterns of interaction, reasoning, and tool use. Unlike ordinary tool APIs, which expose external capabilities, skills often encode higher-level behavioral strategies: how to debug a software failure, navigate a household environment, compare candidate products, collect evidence for a claim, or follow domain-specific conventions. Recent surveys characterize skills as procedural artifacts that coordinate tools, memory, and runtime context under task-specific constraints, bridging the "procedural gap" between raw tool access and robust task execution (Xu and Yan, 2026; Jiang et al., 2026; Zhou et al., 2026). In this sense, skills act as the operational layer of agent systems: they externalize know-how that would otherwise need to be rediscovered from scratch in every task.

The skill-agent ecosystem has expanded rapidly across the full skill lifecycle. Skills can be represented as text-based instructions, code-backed routines, or hybrid artifacts (Shinn et al., 2023; Zhao et al., 2024; Wang et al., 2024b, 2023, 2024a; Zheng et al., 2025); acquired from execution traces, task requirements, or external corpora (Wang et al., 2023; Ni et al., 2026; Qian et al., 2023; Qin et al., 2024); retrieved and selected through dense, generative, or structure-aware mechanisms (Wang et al., 2025, 2026a; Zheng et al., 2026; Liu et al., 2026; Li et al., 2026a); and further compressed, compiled, structured, or evolved as environments change (Gao et al., 2026; Chen et al., 2026; Xia et al., 2026b; Lu et al., 2026; Zhang et al., 2026b). This lifecycle view shows that skills are no longer merely prompt snippets or isolated tools, but persistent procedural objects embedded in a broader ecosystem of representation, acquisition, retrieval, execution, and evolution (Zhou et al., 2026).

However, this lifecycle also reveals a missing dimension. Most work asks how skills should be written, acquired, retrieved, compressed, composed, or updated. These are essential questions, but they do not fully determine how a retrieved skill should affect the agent at inference time. A skill is not automatically useful simply because it is relevant. It may be too long, too vague, too specific, too outdated, too weakly grounded in the current environment, or too high-level for the agent's current state. Conversely, a skill that fails as a full document may still help as a short reminder, a compact affordance-and-risk summary, a workflow template,

*Corresponding author.

Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing, pages 58285–58300
October 24-29, 2026 ©2026 Association for Computational Linguistics

or a compact operational summary. We therefore argue that skills should be viewed as externalized procedural priors whose effect depends not only on their content, but also on the interface through which they condition the agent. The practical question is not only which skill should be selected, but also how the selected skill should be exposed.

In this paper, we propose SKILLALIGN, a framework for skill-interface alignment in LLM agents. SkillAlign separates *skill selection* from *skill exposure*: upstream providers determine which skills are available, while SkillAlign determines how those skills condition the agent. Given candidate skills from any provider, SkillAlign converts each raw skill into a multi-view procedural card and renders the candidate set through an exposure interface, such as a full document, short hint, compressed operational summary, workflow, or no exposure. This turns exposure into a controlled decision variable, allowing us to evaluate whether different interfaces for the same task and candidate skills lead to different trajectories, costs, and failures. SkillAlign further supports learning an exposure policy from counterfactual outcomes, providing a path toward adaptive interface selection.

We evaluate SkillAlign on ALFWorld and SkillsBench under multiple skill providers and agent backends. Our experiments compare fixed exposure modes and learned exposure policies while holding candidate skills fixed when needed. The results show that exposure form changes task success and context cost even when the same skills are selected. In particular, concise exposure can match or approach full-document injection in some settings, while learned selectors reveal substantial headroom over any single fixed interface.

Our contributions are as follows:

- We identify *skill-interface alignment* as a missing problem in skill-augmented agents: skill utility depends not only on skill relevance or quality, but also on how the skill is exposed to the agent.
- We propose SKILLALIGN, an exposure framework that represents skills as multi-view procedural cards and renders candidate skills through alternative interfaces, including full instructions, hints, compressed summaries, workflows, or no exposure.
- We introduce a counterfactual evaluation protocol that holds the task, agent backend, and candidate skills fixed while varying only the exposure interface, enabling direct measurement of interface-induced changes in task success and rendered skill-context cost.
- We evaluate SkillAlign across ALFWorld and SkillsBench with multiple providers and agent backends, showing that exposure form is a first-class factor in skill-augmented performance and that adaptive exposure contains learnable signal.

2 Related Work

2.1 Agent Skills as Procedural Priors

Agent skills externalize reusable procedural knowledge for LLM agents. Recent surveys define skills as modular artifacts that coordinate tools, memory, and runtime context under task-specific constraints, thereby bridging the procedural gap between raw tool access and reliable execution (Xu and Yan, 2026; Jiang et al., 2026; Zhou et al., 2026). This view distinguishes skills from ordinary tools: tools expose operations, while skills package know-how for when and how those operations should be used in context. Skills therefore act as procedural priors that bias the agent toward reusable patterns of reasoning, interaction, and tool use.

Existing systems instantiate such priors in different forms. Text-based skills encode reflections, lessons, reasoning templates, or workflow memories; code-backed skills include executable scripts, API calls, or programmatic routines; and hybrid skills combine textual guidance with executable or structured runtime support (Shinn et al., 2023; Zhao et al., 2024; Yang et al., 2024b; Wang et al., 2024b; Ni et al., 2026; Wang et al., 2023, 2024a; Yang et al., 2024a; Zhang et al., 2023; Zheng et al., 2025). SkillAlign builds on this procedural-prior view but studies a different design variable: not how a skill is acquired or represented, but how an already available skill should condition the agent in a given task state.

2.2 Skill Acquisition, Retrieval, and Structured Use

A large body of work studies how skills are obtained and made available. Skills may be distilled from prior trajectories, generated from current task requirements, extracted from external corpora, or accumulated through lifelong evolution (Wang et al., 2023; Shinn et al., 2023; Zhao et al., 2024; Wang et al., 2024b; Ni et al., 2026; Qian

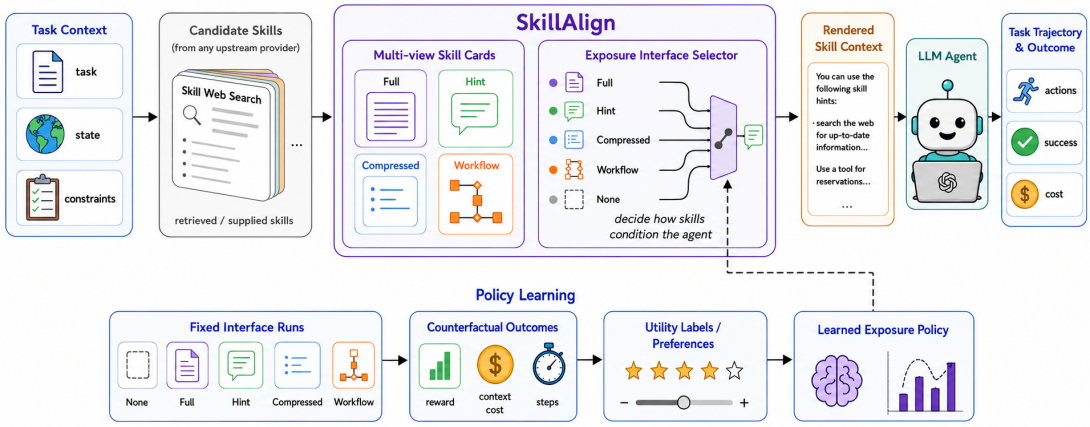

Figure 1: Overview of SkillAlign. Given a task context and candidate skills from any upstream provider, SkillAlign constructs multi-view skill cards and selects an exposure interface that determines how the skills condition the agent. Fixed-interface runs provide counterfactual outcomes for evaluating exposure effects. These outcomes can also supervise an exposure policy that selects interfaces adaptively.

et al., 2023; Qin et al., 2024; Zhang et al., 2026a; Xia et al., 2026a; Yang et al., 2026; Wang et al., 2026b). These works address where skills come from and how they are improved; SkillAlign instead starts after candidate skills are available.

Another line of work studies retrieval, routing, and selection from growing skill libraries (Wang et al., 2023, 2025, 2026a; Zheng et al., 2026; Liu et al., 2026; Li et al., 2026a). Dense and sparse retrieval match task contexts to skill descriptions or metadata, generative retrieval predicts tool or skill identifiers, and structure-aware methods exploit hierarchies, dependencies, or graph relations. Other systems optimize the skill artifact itself (Gao et al., 2026; Chen et al., 2026; Wang et al., 2026c; Lu et al., 2026; Xia et al., 2026b): SkillReducer compresses verbose skills, SkVM compiles skills across model-harness pairs, WebXSkill bridges textual workflow guidance with executable web programs, ContractSkill formalizes executable contracts, and GraSP composes retrieved skills into verified graph-structured procedures . These directions mainly ask which skills should be surfaced, shortened, compiled, or composed. SkillAlign is complementary: it asks how selected skills should be exposed to the agent.

3 Method

SkillAlign focuses on a design choice that is usually implicit in skill-augmented agents: given a set of candidate skills, *through what interface should these skills condition the agent*? As illustrated in Figure 1, SkillAlign takes the task context, agent context, and candidate skills as input. The candidate skills may come from any upstream source, such as exhaustive skill provision, vector retrieval, graph-based retrieval, or oracle annotations. SkillAlign then represents each raw skill as a multi-view procedural card and renders the candidate set through a selected exposure interface, such as a full instruction, short hint, compressed summary, workflow, or no exposure. The rendered skill context is injected into the agent's ordinary interaction loop, and the resulting trajectory is evaluated by the benchmark. Our main analysis uses fixed exposure modes to create counterfactual trajectories, while learned exposure policies use these outcomes to study adaptive interface selection. **Note that**, in this work, the learned policy is intentionally lightweight: it is used to validate the feasibility of adaptive exposure rather than to fully solve interface selection. We therefore report it as a policy-learning analysis and compare it against oracle exposure to quantify the remaining headroom.

3.1 Skill Exposure as an Interface Decision

Let x denote the task context, a the agent context, and $S_x = \{s_1, \ldots, s_k\}$ the candidate skills available for the task. SkillAlign treats S_x as given: the candidate set may come from exhaustive skill provision, vector retrieval, graph-based retrieval, or oracle annotations, but retrieval itself is not our object of optimization. Instead, SkillAlign chooses an exposure interface m:

$$m = \pi(x, a, S_x), \qquad m \in \mathcal{M},$$

and renders the same candidate skills into an agent-facing context:

$$r = \mathrm{Render}(S_x, m; x, a).$$

The rendered context r does not replace or modify the original skill artifact. It is the interface through which the skill conditions the agent at inference time. In our main experiments,

$$\mathcal{M} = \{\texttt{none}, \texttt{full}, \texttt{hint}, \texttt{compressed}, \texttt{workflow}\}.$$

These interfaces define different strengths and structures of conditioning. `full` exposes the original skill document; `hint` gives a concise task-facing cue; `compressed` preserves key affordances, assumptions, constraints, and risks; `workflow` emphasizes ordered procedural guidance; and `none` suppresses skill text when candidate skills are unnecessary or potentially harmful. This formulation makes exposure a controlled variable: for the same task, agent, and candidate skills, different interfaces may induce different trajectories.

3.2 Multi-View Skill Cards and Rendering

Raw skills are heterogeneous: they may be natural-language instructions, reflections, workflow memories, scripts, API recipes, repository-specific procedures, or hybrid packages. SkillAlign represents each skill as a *multi-view skill card*, preserving the original artifact while exposing alternative task-facing views (see Appendix B):

$$c(s) = \{v_{\text{full}}, v_{\text{hint}}, v_{\text{comp}}, v_{\text{flow}}, \eta\},$$

where v_{full} is the original skill content, v_{hint} is a concise cue, v_{comp} is a compact affordance-and-risk view, v_{flow} is an ordered workflow view, and η stores metadata such as skill id, provenance, source path, and retrieval-facing summary.

The views differ semantically rather than merely in length. The full view preserves execution-critical details such as assumptions, examples, scripts, and edge cases. The hint view acts as a weak activation signal. The compressed view summarizes applicability, actions, expected outputs, and mismatch risks. The workflow view emphasizes ordering and checkpoints. Given an interface m, the renderer selects the corresponding view and constructs the injected context, framing skills as supportive priors rather than instructions that override task constraints or environment feedback.

3.3 Fixed Exposure Modes and Counterfactual Evaluation

The main evidence for SkillAlign comes from fixed exposure modes. A fixed policy always chooses the same interface,

$$\pi_m(x, a, S_x) = m,$$

and for each task and candidate source, we run the agent under multiple interfaces over the same candidate skill set:

$$\{\tau_x^m : m \in \mathcal{M}\}.$$

These counterfactual trajectories isolate the effect of exposure form. If the task, agent backend, and candidate skills are held fixed, but performance changes across `full`, `hint`, `compressed`, `workflow`, and `none`, then the interface is not a cosmetic formatting choice but a substantive factor in skill utility.

We use these trajectories to measure task success or reward and injected context cost. A task is counted as a negative-transfer case for interface m if the no-exposure run succeeds but the run with m fails, or if scalar reward decreases under m. Repeating the same comparison under different candidate sources allows us to analyze how exposure interacts with upstream skill provision without making retrieval itself the contribution.

3.4 Exploratory Exposure Policy Learning

We further study whether exposure choice can be learned from counterfactual outcomes. This component is not required by the SkillAlign framework, but it provides a way to estimate how much interface selection can be improved beyond fixed exposure modes.

For each task and candidate set, fixed-mode runs provide outcomes under all interfaces from an auxiliary dataset. We convert them into a utility score:

$$U_x(m) = R_x(m) - \alpha C_x(m) - \beta T_x(m),$$

where $R_x(m)$ is reward or success, $C_x(m)$ is normalized context cost, and $T_x(m)$ is normalized trajectory length. The best-utility interface gives an SFT label,

$$m_x^* = \arg\max_{m \in \mathcal{M}} U_x(m),$$

while interface pairs with sufficient utility margin form preference examples for DPO. The

learned policy receives the task description, candidate skill-card content, and source metadata, then predicts a single interface label for the candidates.

In our experiments, we train lightweight LoRA adapters on Qwen3-4B model (Yang et al., 2025). SFT serves as a stable first attempt at learning exposure decisions, while DPO tests whether preference optimization over counterfactual utilities can further improve interface choice. Note that the current exposure policy learning is deliberately basic and lightweight, intended to verify the feasibility of optimizing skill interfaces rather than to solve the problem fully. More details are in Appendix E.

4 Experiments

We evaluate whether skill exposure is a first-class factor in skill-augmented agents. The central question is: given the same task, agent backend, and candidate skill set, does changing the exposure interface alter task success or context cost? To answer this question, we conduct controlled counterfactual evaluations across interactive and tool-use agent benchmarks. For each candidate source, we render the same skills through different interfaces—`full`, `hint`, `compressed`, `workflow`, or `none`—and compare the resulting trajectories. This design directly tests our hypothesis that skills act as externalized procedural priors whose effect depends on how they condition the agent.

4.1 Setup

Benchmarks. We evaluate on two benchmarks for skill-augmented agents. **ALFWorld** (Shridhar et al., 2020) is a text-based embodied environment where agents complete multi-step household tasks by observing textual states and issuing textual actions. Following ReAct-style evaluation (Yao et al., 2022), we use the text-only setting with 140 evaluation episodes. **SkillsBench** (Li et al., 2026b) contains real-world technical tasks paired with curated skills, covering domains such as data analysis, scientific computing, financial modeling, and engineering workflows. We evaluate on 94 SkillsBench task instances using Harbor and `Terminus-2` (Harbor Framework Team, 2026) execution harness. We use **ScienceWorld** (Wang et al., 2022) for auxiliary exposure-policy data generation.

Skill libraries. We denote a skill library as `skills_N`, where N is the number of available skills, not the number of evaluation tasks. The main experiments use `skills_1000`. For scaling analysis, we additionally evaluate `skills_200`, `skills_500`, and `skills_2000`. Skill cards are constructed from raw skill artifacts before evaluation. For overlapping skill ids with identical raw content, cards are reused across libraries.

Candidate sources. SkillAlign is provider-agnostic: it operates on candidate skills supplied by upstream sources. The main experiments use three generic candidate sources: `none`, which injects no skill context; `all`, which makes the available skill library available without task-specific filtering; and `vector-top`k, which retrieves skills using semantic similarity via `text-embedding-3-large` embeddings. We use $k = 5$ for ALFWorld and $k = 8$ for SkillsBench. These sources represent no exposure, broad skill availability, and standard top-k retrieval. We additionally evaluate a graph-based provider in Section 4.4 to test whether exposure effects persist under structured skill selection.

Exposure interfaces. For each candidate source, we compare five exposure interfaces: `none`, `full`, `hint`, `compressed`, and `workflow`. The `full` interface injects the original skill content; `hint` provides concise task-facing cues; `compressed` summarizes affordances, assumptions, constraints, and risks; `workflow` presents ordered procedural guidance; and `none` suppresses skill text.

Agent backends and metrics. We evaluate three backends: `minimax-m2.7`, `gpt-5.3-codex`, and `glm-5`, accessed through API services. We report success rate (SR) or average reward (AR), together with rendered skill-context cost C. For ALFWorld, reward is binary and equals success rate. C denotes the token length of the rendered skill context, reported in thousands of tokens. For overlong all-skill settings, C reflects the rendered exposure scale before backend-specific truncation or rejection.

4.2 Main Results

The main comparison asks whether exposure form changes agent behavior when the candidate skills are held fixed. Table 1 reports both performance and rendered skill-context cost under `skills_1000` across three agent backends. Rows specify the candidate source and exposure interface, while columns report performance–cost pairs for each backend.

Exposure is a first-class conditioning variable. Across benchmarks and backends, changing only

Configuration	ALFWorld						SkillsBench					
	MiniMax		GPT-Codex		GLM		MiniMax		GPT-Codex		GLM	
	SR ↑	C ↓	SR ↑	C ↓	SR ↑	C ↓	AR ↑	C ↓	AR ↑	C ↓	AR ↑	C ↓
No Skill / None	50.7	0.0	89.3	0.0	73.6	0.0	17.2	0.0	27.4	0.0	22.6	0.0
All + Full	47.9	900.0	88.6	900.0	70.7	900.0	14.5	1080.0	24.8	1080.0	20.4	1080.0
All + Hint	68.6	6.0	92.1	6.0	80.7	6.0	16.8	7.5	29.4	7.5	24.0	7.5
All + Compressed	72.1	25.2	93.6	25.2	82.1	25.2	17.2	31.0	30.8	31.0	25.1	31.0
All + Workflow	61.4	282.5	90.0	282.5	76.4	282.5	15.6	335.0	27.6	335.0	22.8	335.0
Vector-top$k$ + Full	73.6	4.7	92.9	4.7	80.7	4.7	16.4	9.0	31.8	9.0	24.8	9.0
Vector-top$k$ + Hint	**75.0**	0.08	**94.3**	0.08	**83.6**	0.08	15.6	0.13	28.9	0.13	23.4	0.13
Vector-top$k$ + Compressed	69.3	0.30	94.0	0.30	82.9	0.30	**18.1**	0.55	**32.2**	0.55	**26.2**	0.55
Vector-top$k$ + Workflow	64.3	1.5	91.4	1.5	78.6	1.5	16.2	2.9	29.6	2.9	24.4	2.9

Table 1: Main SkillAlign results under `skills_1000`. ALFWorld reports success rate (SR), and SkillsBench reports average reward (AR). C denotes the token length of the rendered skill context, reported in thousands of tokens. Each row fixes a candidate source and exposure interface; within each candidate-source block, candidate skills are fixed and only the exposure interface changes. Best performance results are bolded and second-best results are underlined within each benchmark–backend block. Task-level bootstrap 95% confidence intervals are reported in Appendix C.1. Lower cost is better.

the exposure interface leads to substantial performance differences. Under the same `all` candidate source, ALFWorld success varies from 47.9 to 72.1 for MiniMax and from 70.7 to 82.1 for GLM. Under `vector-top`k, the same candidate set also yields different outcomes across interfaces, such as 64.3–75.0 on ALFWorld with MiniMax and 28.9–32.2 on SkillsBench with GPT-Codex. This supports our claim that a skill is not simply useful or harmful in isolation; it becomes useful through an interface that controls how strongly and in what form it conditions the agent.

Candidate quality and exposure form interact. Although SkillAlign does not propose a new retriever, the results show that exposure effects depend on the quality of the candidate set. `All + Compressed` improves over `All + Full`, but it still aggregates compressed information from all 1000 skills and can contain many irrelevant procedural priors. In contrast, `Vector-top`k substantially reduces the candidate set before exposure. This pattern shows that candidate filtering and exposure design are complementary: retrieval reduces irrelevant skill mass, while SkillAlign controls how the remaining procedural priors are presented.

Compact exposure is often more efficient. Compact interfaces often achieve strong reward at much lower cost. For example, `Vector-top`k `+ Hint` and `Vector-top`k `+ Compressed` achieve top-ranking results in multiple columns while remaining below 1K token-equivalent units for compact views. Among them, `hint` is the shortest interface, while `compressed` uses more context to

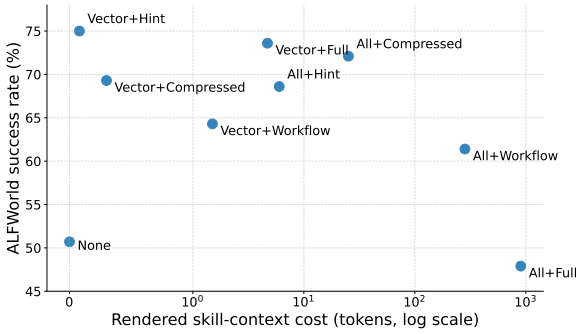

Figure 2: Cost–performance tradeoff of exposure interfaces under MiniMax M2.7 on ALFWorld. Each point corresponds to a row in the ALFWorld–MiniMax block of Table 1. Interfaces closer to the upper-left provide better utility per unit of rendered skill-context cost.

preserve triggers, state assumptions, actions, outputs, and risks. This explains why `compressed` can outperform `hint` on more technical tasks on SkillsBench despite using slightly more context.

All + Full as a diagnostic condition. `All + Full` performs worse than several compact alternatives, but we do not attribute this degradation to a single mechanism. It may reflect a combination of backend truncation or rejection, attention dilution, irrelevant procedural priors, and instruction conflict among many skills. Disentangling these factors would require dedicated controls over context length, relevance, and ordering. We therefore treat `All + Full` primarily as a diagnostic stress condition and place greater emphasis on the `Vector-top`k and `Graph-top`k comparisons, where candidate skills are already filtered.

Configuration	ALFWorld						SkillsBench					
	MiniMax		GPT-Codex		GLM		MiniMax		GPT-Codex		GLM	
	SR ↑	C ↓	SR ↑	C ↓	SR ↑	C ↓	AR ↑	C ↓	AR ↑	C ↓	AR ↑	C ↓
Graph-top$k$ + Full	66.4	4.4	93.6	4.4	80.0	4.4	17.9	5.3	32.8	5.3	25.4	5.3
Graph-top$k$ + Hint	68.6	0.07	92.1	0.07	76.4	0.07	16.3	0.12	30.5	0.12	23.9	0.12
Graph-top$k$ + Compressed	**71.4**	0.28	**95.0**	0.28	**84.3**	0.28	**19.0**	0.48	**34.6**	0.48	**27.1**	0.48
Graph-top$k$ + Workflow	63.6	1.4	91.4	1.4	75.7	1.4	17.4	1.7	31.5	1.7	24.6	1.7

Table 2: Structured-provider analysis under `skills_1000`. `Graph-top`k uses Graph-of-Skills (Liu et al., 2026) as a representative graph-based provider. C denotes the token length of the rendered skill context, reported in thousands of tokens. The results show that interface choice remains important even with different skill selectors or providers. Task-level bootstrap 95% confidence intervals are reported in Appendix C.1.

4.3 Cost–Performance Tradeoff

A central motivation for SkillAlign is that exposure affects both utility and context cost. Figure 2 plots the MiniMax M2.7 cost–performance tradeoff on ALFWorld under the generic candidate sources in Table 1. We use this setting because its values directly correspond to one benchmark–backend block in the main table. Interfaces in the upper-left region achieve higher success with lower exposure cost.

Full exposure applies a strong but expensive procedural prior. Hint and compressed exposure apply weaker or more structured priors at much lower cost. When these compact interfaces achieve comparable or better success, exposing less skill content is not merely an efficiency trick; it is a better alignment between the agent's need and the skill's influence.

4.4 Influence of Candidate Provider

The main experiments use generic candidate sources to avoid tying SkillAlign to a specific retrieval system. To test whether the exposure effect persists under structured skill selection, we additionally evaluate a graph-based candidate provider as shown in Table 2. We refer to this setting as `Graph-top`k, instantiated with Graph-of-Skills (Liu et al., 2026). This analysis checks whether the same exposure problem remains when candidate skills are selected by a structured provider.

The structured-provider results are consistent with the main findings. `Graph-top`k `+ Compressed` is generally stronger than other graph-provider interfaces, while `Graph-top`k `+ Full` is more costly and not consistently better. This suggests that better candidate selection does not eliminate the exposure problem. Even when a structured provider supplies a compact and relevant candidate set, the agent still benefits from receiving those skills through an aligned interface.

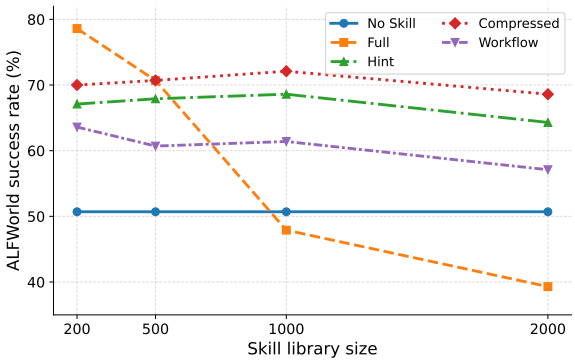

Figure 3: Reward trend under growing skill libraries. The no-skill baseline remains constant because no skill content is exposed. The 500-skill setting is slightly lower than the 1,000-skill setting for compact exposure, while 2,000 skills introduces more noise and cost.

4.5 Scaling with Skill-Library Size

We evaluate whether exposure choice becomes more important as skill libraries grow. The evaluation task set is fixed, while the number of available skills varies from `skills_200` to `skills_500`, `skills_1000`, and `skills_2000`. We conduct this analysis on ALFWorld with MiniMax M2.7 under the all-candidate source, where library size directly controls the amount and diversity of available skill content. Figure 3 visualizes the reward trend across exposure interfaces.

The scaling results make the exposure problem more explicit. At smaller library sizes, `full` exposure can still be competitive because the raw skill context remains manageable. As the library grows, however, `full` exposure becomes increasingly expensive and degrades sharply, dropping to 47.9 at `skills_1000`. Compact interfaces, especially `compressed`, remain more stable and reach 72.1 at `skills_1000`. The 500-skill setting is slightly lower than the 1,000-skill setting for compact exposure, while performance drops again at 2,000

Selector	ALFWorld SR (%)	
	Vector-top$k$	Graph-top$k$
Best Fixed Interface	75.0	71.4
SFT Policy	76.4	73.6
+DPO Policy	78.6	76.4
Oracle Exposure	**92.9**	**85.7**

Table 3: Exploratory exposure policy learning on ALFWorld. SFT and DPO policies are trained from auxiliary counterfactual outcomes and evaluated by replay. Oracle exposure estimates the headroom of perfect task-level interface selection and is not a deployable policy.

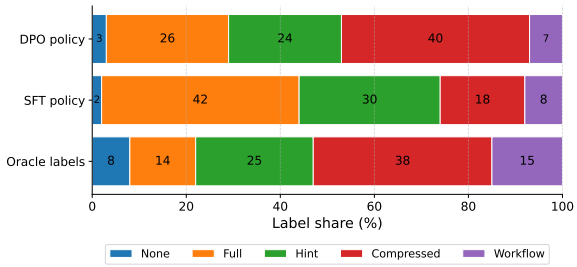

Figure 4: Distribution of exposure-interface labels. Oracle labels are diverse, indicating that no single exposure form is universally optimal. Learned policies partially recover this pattern but remain biased, explaining the remaining gap to oracle exposure.

skills as the available skill pool becomes larger and noisier. These trends support the central motivation of SkillAlign: large skill ecosystems do not only require better retrieval, but also better control over how selected or available skills condition the agent.

4.6 Exploratory Exposure Policy Learning

We further test whether exposure choice can be learned from counterfactual supervision. The training data is generated from auxiliary ScienceWorld, where the same task/provider setting is evaluated under different fixed interfaces. We then evaluate the learned policy on ALFWorld through off-policy replay: the policy selects an interface for each task, and we look up the corresponding fixed-interface trajectory result. This experiment is intended as a feasibility probe rather than the core contribution.

Table 3 compares the best fixed interface, SFT policy, DPO policy, and oracle exposure under two candidate sources. Oracle exposure selects the best realized interface per task from fixed-mode counterfactuals and serves only as an analysis upper bound. Figure 4 compares the distribution of oracle labels and learned policy predictions.

Under replay evaluation, the learned policies improve over the best fixed interface, suggesting that exposure choice contains learnable signal, while also making clear that the current SFT/DPO setup is only a preliminary validation of feasibility. DPO improves over SFT in our replay evaluation, providing preliminary evidence that preference supervision can capture additional exposure-selection signal. However, the large oracle gap and the mismatch in label distributions indicate that more advanced policy learning remains an important direction for future work.

View	Omission	Dist./Contr.	MA.
Hint	2/60 (3.3%)	1/60 (1.7%)	1/60 (1.7%)
Comp.	3/60 (5.0%)	2/60 (3.3%)	1/60 (1.7%)
Workflow	4/60 (6.7%)	2/60 (3.3%)	2/60 (3.3%)

Table 4: Human fidelity audit of generated SkillAlign views. (Comp.: Compressed; Dist.: distortions; Contr.: contradictions; MA.: Misleading additions.)

4.7 Analysis

Fidelity of generated views. Since Hint, Compressed, and Workflow are generated from the raw skill, an observed failure could in principle arise from information omitted or distorted during card generation rather than from the exposure interface itself. We therefore audit 180 task–skill–view instances from 60 randomly sampled tasks (30 from ALFWorld and 30 from SkillsBench) using two independent human annotators. To avoid treating legitimate compression as an error, we count an omission only when information necessary for applying the skill to the paired task is absent. Table 4 summarizes the results. Task-relevant omissions occur in 3.3%, 5.0%, and 6.7% of the Hint, Compressed, and Workflow instances, respectively, while distortions, contradictions, and misleading additions are similarly rare.

Overall, the audit suggests that potential execution- or safety-relevant fidelity failures are uncommon under the current generation mechanism, but they are not completely eliminated. For safety-sensitive applications, compact views can additionally be paired with fidelity verification and fallback to the Full view when necessary.

Generic length-matched summarization. To separate exposure structure from context shortening, we further compare Compressed against a generic free-form summary under both Vector-top-k and Graph-top-k, across all three backends. Generic Summary uses the same candi-

date skills, generation model, decoding settings, and approximately matched rendered length as Compressed. Across the resulting 12 benchmark–provider–backend settings, Compressed generally outperforms Generic Summary at a comparable context budget, while Full remains competitive or stronger in some settings. This indicates that context shortening explains part, but not all, of the benefit of compact exposure, and further supports our conclusion that the appropriate interface is setting-dependent. See full results in Appendix C.2.

Candidate source and interface. Retrieval quality and exposure form are complementary. Top-k retrieval reduces irrelevant skill mass, while SkillAlign controls how the selected skills condition the agent. This explains why compact vector- or graph-based exposure can outperform both no-skill and full-library exposure.

Adaptive exposure. The oracle-policy gap shows that interface choice contains learnable signal, but our simple SFT/DPO policies only recover part of it. This suggests that adaptive exposure is promising, while stronger policy learning is needed to approach oracle-level interface selection.

5 Conclusion

We presented SkillAlign, a framework for studying and improving how skills are exposed to LLM agents. Rather than treating skills as static prompt snippets or assuming that more skill content is always better, SkillAlign views skills as externalized procedural priors whose effect depends on the interface through which they condition the agent. We instantiate this idea by reframing raw skills into multiple exposure interfaces, including full descriptions, hints, compressed summaries, and workflow-style guidance.

Across ALFWorld and SkillsBench, our experiments show that exposure form is a first-class factor in skill-augmented agents. Full-library exposure can be harmful despite containing more information, while compact interfaces often achieve better cost–performance tradeoffs. We further show that exposure effects persist under different candidate providers, and that the best interface varies across tasks, backends, and skill sources. Finally, exploratory policy-learning results indicate that exposure choice is learnable, although current SFT/DPO policies remain far below oracle exposure. These findings suggest that future skill-agent systems should optimize not only skill acquisition and retrieval, but also the interface through which skills are presented to agents.

Limitations

Our work has several limitations.

First, SkillAlign currently relies on a small set of manually designed exposure interfaces. Although these interfaces are sufficient to reveal the importance of skill exposure, they do not cover the full design space of possible skill representations. Future work could learn richer interfaces or generate task-adaptive views directly.

Second, our exposure-policy learning is intentionally simple. The SFT and DPO policies are used mainly as feasibility probes, and the large gap to oracle exposure shows that adaptive exposure remains far from solved. Stronger policy models, better counterfactual data construction, and more robust preference filtering may further improve interface selection.

Third, our framework is complementary to skill retrieval rather than a replacement for it. The quality of candidate skills still matters: poor retrieval can limit the benefit of even a well-aligned exposure interface. This suggests that skill-agent systems should jointly consider retrieval quality, exposure form, and downstream agent behavior.

Use of AI Assistants

The authors used AI assistants (Claude Code and OpenAI Codex) for limited language editing, grammar checking, and code-debugging assistance. All research ideas, method design, experimental decisions, experiment execution, result analysis, and manuscript writing were conducted by the authors. The authors reviewed all AI-assisted outputs and take full responsibility for the final content.

Acknowledgements

This work is supported by the Strategic Priority Research Program of Chinese Academy of Sciences under Grant XDA04080400 and Beijing Natural Science Foundation L259016.

References

Le Chen, Erhu Feng, Yubin Xia, and Haibo Chen. 2026. Skvm: Revisiting language vm for skills across heterogenous llms and harnesses. *arXiv preprint arXiv:2604.03088*.

Yudong Gao, Zongjie Li, Yuanyuan Yuan, Zimo Ji, Pingchuan Ma, and Shuai Wang. 2026. Skillreducer: Optimizing llm agent skills for token efficiency. *arXiv preprint arXiv:2603.29919*.

Harbor Framework Team. 2026. Harbor: A framework for evaluating and optimizing agents and models in container environments.

Yanna Jiang, Delong Li, Haiyu Deng, Baihe Ma, Xu Wang, Qin Wang, and Guangsheng Yu. 2026. Sok: Agentic skills–beyond tool use in llm agents. *arXiv preprint arXiv:2602.20867*.

Hao Li, Chunjiang Mu, Jianhao Chen, Siyue Ren, Zhiyao Cui, Yiqun Zhang, Lei Bai, and Shuyue Hu. 2026a. Organizing, orchestrating, and benchmarking agent skills at ecosystem scale. *arXiv preprint arXiv:2603.02176*.

Xiangyi Li, Yimin Liu, Wenbo Chen, Bingran You, Zonglin Di, Yifeng He, Shenghan Zheng, Kyoung Whan Choe, Jiankai Sun, Shuyi Wang, and et al. 2026b. Skillsbench: Benchmarking how well agent skills work across diverse tasks. *arXiv preprint arXiv:2602.12670*.

Dawei Liu, Zongxia Li, Hongyang Du, Xiyang Wu, Shihang Gui, Yongbei Kuang, and Lichao Sun. 2026. Graph-of-skills: Dependency-aware structural retrieval for massive agent skills. *arXiv preprint arXiv:2604.05333*.

Zijian Lu, Yiping Zuo, Yupeng Nie, Xin He, Weibei Fan, Lianyong Qi, and Shi Jin. 2026. Contractskill: Repairable contract-based skills for multimodal web agents. *arXiv preprint arXiv:2603.20340*.

Jingwei Ni, Yihao Liu, Xinpeng Liu, Yutao Sun, Mengyu Zhou, Pengyu Cheng, Dexin Wang, Erchao Zhao, Xiaoxi Jiang, and Guanjun Jiang. 2026. Trace2skill: Distill trajectory-local lessons into transferable agent skills. *arXiv preprint arXiv:2603.25158*.

Cheng Qian, Chi Han, Yi Fung, Yujia Qin, Zhiyuan Liu, and Heng Ji. 2023. Creator: Tool creation for disentangling abstract and concrete reasoning of large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 6922–6939.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, and et al. 2024. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *International Conference on Learning Representations*, volume 2024, pages 9695–9717.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2020. Alfworld: Aligning text and embodied environments for interactive learning.

Fali Wang, Chenglin Weng, Xianren Zhang, Siyuan Hong, Hui Liu, and Suhang Wang. 2026a. Graphskill: Documentation-guided hierarchical retrieval-augmented coding for complex graph reasoning. *arXiv preprint arXiv:2603.06620*.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*.

Jiongxiao Wang, Qiaojing Yan, Yawei Wang, Yijun Tian, Soumya Smruti Mishra, Zhichao Xu, Megha Gandhi, Panpan Xu, and Lin Lee Cheong. 2026b. Reinforcement learning for self-improving agent with skill library. pages 1529–1550.

Renxi Wang, Xudong Han, Lei Ji, Shu Wang, Timothy Baldwin, and Haonan Li. 2025. Toolgen: Unified tool retrieval and calling via generation. In *International Conference on Learning Representations*, volume 2025, pages 73473–73498.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. 2022. Scienceworld: Is your agent smarter than a 5th grader? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11279–11298.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. 2024a. Executable code actions elicit better llm agents.

Zhaoyang Wang, Qianhui Wu, Xuchao Zhang, Chaoyun Zhang, Wenlin Yao, Fazle Elahi Faisal, Baolin Peng, Si Qin, Suman Nath, Qingwei Lin, and et al. 2026c. Webxskill: Skill learning for autonomous web agents. *arXiv preprint arXiv:2604.13318*.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. 2024b. Agent workflow memory. *arXiv preprint arXiv:2409.07429*.

Peng Xia, Jianwen Chen, Hanyang Wang, Jiaqi Liu, Kaide Zeng, Yu Wang, Siwei Han, Yiyang Zhou, Xujiang Zhao, Haifeng Chen, and et al. 2026a. Skillrl: Evolving agents via recursive skill-augmented reinforcement learning. *arXiv preprint arXiv:2602.08234*.

Tianle Xia, Lingxiang Hu, Yiding Sun, Ming Xu, Lan Xu, Siying Wang, Wei Xu, and Jie Jiang. 2026b. Grasp: Graph-structured skill compositions for llm agents. *arXiv preprint arXiv:2604.17870*.

Renjun Xu and Yang Yan. 2026. Agent skills for large language models: Architecture, acquisition, security, and the path forward. *arXiv preprint arXiv:2602.12430*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and et al. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

John Yang, Carlos Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. 2024a. Swe-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems*, 37:50528–50652.

Ling Yang, Zhaochen Yu, Tianjun Zhang, Shiyi Cao, Minkai Xu, Wentao Zhang, Joseph E Gonzalez, and Bin Cui. 2024b. Buffer of thoughts: Thought-augmented reasoning with large language models. *Advances in Neural Information Processing Systems*, 37:113519–113544.

Yutao Yang, Junsong Li, Qianjun Pan, Bihao Zhan, Yuxuan Cai, Lin Du, Jie Zhou, Kai Chen, Qin Chen, Xin Li, and et al. 2026. Autoskill: Experience-driven lifelong learning via skill self-evolution. *arXiv preprint arXiv:2603.01145*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models.

Haozhen Zhang, Quanyu Long, Jianzhu Bao, Tao Feng, Weizhi Zhang, Haodong Yue, and Wenya Wang. 2026a. Memskill: Learning and evolving memory skills for self-evolving agents. *arXiv preprint arXiv:2602.02474*.

Kechi Zhang, Huangzhao Zhang, Ge Li, Jia Li, Zhuo Li, and Zhi Jin. 2023. Toolcoder: Teach code generation models to use api search tools. *arXiv preprint arXiv:2305.04032*.

Ziao Zhang, Kou Shi, Shiting Huang, Avery Nie, Yu Zeng, Yiming Zhao, Zhen Fang, Qishen Su, Haibo Qiu, Wei Yang, and et al. 2026b. Skillflow: Benchmarking lifelong skill discovery and evolution for autonomous agents. *arXiv preprint arXiv:2604.17308*.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. 2024. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19632–19642.

Boyuan Zheng, Michael Y Fatemi, Xiaolong Jin, Zora Zhiruo Wang, Apurva Gandhi, Yueqi Song, Yu Gu, Jayanth Srinivasa, Gaowen Liu, Graham Neubig, and et al. 2025. Skillweaver: Web agents can self-improve by discovering and honing skills. *arXiv preprint arXiv:2504.07079*.

YanZhao Zheng, ZhenTao Zhang, Chao Ma, YuanQiang Yu, JiHuai Zhu, Yong Wu, Tianze Xu, Baohua Dong, Hangcheng Zhu, Ruohui Huang, and et al. 2026. Skillrouter: Skill routing for llm agents at scale. *arXiv preprint arXiv:2603.22455*.

Yingli Zhou, Wang Shu, Yaodong Su, Wenchuan Du, Yixiang Fang, and Xuemin Lin. 2026. A comprehensive survey on agent skills: Taxonomy, techniques, and applications. *arXiv preprint arXiv:2605.07358*.

A Implementation and Evaluation Details

A.1 Provider-Agnostic Exposure Layer

SkillAlign is implemented as a provider-agnostic exposure layer. Benchmarks and upstream providers supply task context, agent context, and candidate skills; SkillAlign decides how the candidate skills are rendered before they are inserted into the agent prompt. This separation keeps the exposure layer independent of benchmark-specific runners and retrieval systems. In particular, the core abstraction only assumes generic task, agent, and candidate-skill objects, rather than ALFWorld-, SkillsBench-, ScienceWorld-, or graph-retrieval-specific classes.

The output of SkillAlign is a rendered skill context together with metadata used for analysis. This metadata includes the selected skill ids, exposure interface, rendered length, and decision records. Since the same candidate skills can be rendered through different interfaces, the implementation naturally supports counterfactual comparisons where only exposure form changes.

A.2 Candidate Providers

SkillAlign separates candidate provision from exposure selection. In the experiments, we use three generic candidate sources and one structured provider for analysis ($k = 5$ for ALFWorld and $k = 8$ for SkillsBench):

- `none`: no skill context is provided, serving as the no-exposure control.
- `all`: the full skill library is passed to SkillAlign, isolating exposure effects from retrieval.
- `vector-top`k: skills are retrieved through semantic similarity over cached skill representations.
- `Graph-top`k: a representative graph-based provider used only to test whether exposure effects persist under structured candidate selection.

The provider identity is logged separately from the exposure interface. This allows us to distinguish failures caused by poor candidate selection from failures caused by inappropriate exposure.

Setting	Description
Main benchmarks	ALFWorld and SkillsBench.
Auxiliary data	ScienceWorld and earlier TextWorld runs are used for exposure-policy data construction.
Main skill library	`skills_1000`.
Library-size analysis	`skills_200`, `skills_500`, `skills_1000`, and `skills_2000`.
ALFWorld episodes	140 evaluation episodes with a maximum of 30 steps.
SkillsBench tasks	94 containerized technical tasks.
Backends	`minimax-m2.7`, `gpt-5.3-codex`, and `glm-5`.
Interfaces	`none`, `full`, `hint`, `compressed`, and `workflow`.

Table 5: Summary of the main SkillAlign evaluation settings.

Field	Role
`skill_id`	Unique skill identifier.
`name`	Human-readable skill name.
`description`	Short factual description.
`routing_summary`	Use/avoid boundary for deciding whether the skill belongs in context.
`short_hint`	One immediate action cue.
`compressed`	Compact operational specification.
`workflow`	Ordered procedure with checks and stopping conditions.
`raw_text`	Original full skill document.

Table 6: SkillAlign card schema.

A.3 Benchmark Settings

Table 5 summarizes the main evaluation settings.

For each run, we log the task id, provider, selected skill ids, exposure interface, rendered context length, task reward or success, number of steps when available, backend model, and experiment metadata. These logs allow us to reconstruct cost–performance tradeoffs, interface distributions, negative-transfer cases, and oracle exposure results from fixed-interface trajectories.

B Skill Cards and Exposure Interfaces

B.1 Skill Card Schema

A SkillAlign card stores multiple views of the same underlying skill. These views are designed for different cognitive roles, not merely for different lengths. Table 6 summarizes the main fields.

The `full` interface renders `raw_text`; it is not an LLM-generated summary. This distinction is important because raw skills may contain execution-critical scripts, arguments, assumptions, examples, APIs, or configuration details. SkillAlign may summarize such information in the `compressed` or `workflow` views, but it does not modify the original skill artifact.

Interface	Role
`none`	Exposes no skill text. Used when candidate skills are irrelevant, misleading, or too costly.
`full`	Exposes the raw skill document. Useful when exact scripts, APIs, arguments, or implementation details are needed.
`hint`	Exposes one short action-oriented cue. Useful when the agent only needs a weak nudge.
`compressed`	Exposes trigger conditions, inputs/state, core action, expected output, and risks.
`workflow`	Exposes ordered steps with verification points and stopping conditions. Useful when sequencing matters.

Table 7: SkillAlign exposure interfaces and their intended roles.

B.2 Exposure Interface Roles

Table 7 summarizes the five primary exposure interfaces.

The legacy `checklist` field is retained only for backward compatibility and is not used as a primary experimental mode. Its checkpoint content is folded into the `workflow` view.

B.3 Example Skill Card

Table 8 gives a compact example for an ALFWorld inventory-management skill. The example illustrates how different views expose different information rather than simply shortening the same text.

C Supplementary Analysis

C.1 Uncertainty Analysis

We estimate uncertainty for all performance results in Tables 1 and 2 using task-level bootstrap resampling. For each configuration, we resample the evaluation tasks with replacement 10,000 times and recompute the success rate (ALFWorld) or average reward (SkillsBench). We report percentile-based 95% confidence intervals in Table 9. These intervals quantify uncertainty across evaluation tasks rather than backend run-to-run stochasticity.

Overall, task resampling preserves the main qualitative patterns of the point estimates, while closely spaced results often have substantially overlapping

Field	Example Content
skill_id	alfworld-inventory-management
routing_summary	Select for ALFWorld tasks where the agent must track carried objects, containers, receptacles, or object placement across rooms. Avoid for pure navigation when no object state must be remembered.
short_hint	Check inventory after every pickup and before every put/open/heat/cool action.
compressed	Trigger: the task requires carrying, placing, heating, cooling, cleaning, or inspecting objects. Inputs/State: current observation, inventory, target object, receptacles, and appliance states. Do: maintain an explicit inventory and location note; verify object possession before acting on it; verify receptacle or appliance state before placing or transforming objects. Output: fewer invalid actions and clearer goal progress. Watch: using objects not in inventory, forgetting object locations, or acting before opening or activating containers.
workflow	Observe current room, visible target objects, candidate receptacles, and inventory. Before pickup, ensure the object is visible and reachable. Before put, heat, cool, clean, or examine actions, verify that the needed object is currently carried. After placing an object, remember the receptacle location and state. Stop when the environment gives success or the goal condition is visibly satisfied.

Table 8: Compact example of a SkillAlign card for an embodied-agent skill.

confidence intervals. Such small numerical differences should therefore not be over-interpreted. This is consistent with our main conclusion that no single exposure interface is universally superior; instead, the effect of exposure depends on the task, backend, and candidate provider.

C.2 Generic Length-matched Summarization

To distinguish the effect of exposure structure from context shortening, we compare the structured Compressed interface against a generic free-form summarization baseline in Table 10. We evaluate this baseline on the full ALFWorld and SkillsBench evaluation sets under both `Vector-top`k and `Graph-top`k, using all three agent backends. For each setting, Full, Generic Summary, and Compressed use the same candidate skill ids and ordering. Generic Summary uses the same generation model and decoding settings as Compressed, with approximately matched rendered context length.

At comparable compact context lengths, Compressed generally outperforms Generic Summary across providers, backends, and benchmarks, while Full remains stronger in some settings. These results suggest that context shortening explains part, but not all, of the benefit of compact exposure. More broadly, they are consistent with our central finding that the appropriate exposure interface is setting-dependent, rather than universally fixed.

D Prompt Templates

D.1 Skill Card Generation

The LLM-backed card generator is instructed to produce faithful structured views of the source skill. The prompt emphasizes that the views should play different cognitive roles and that execution contracts should be preserved when present.

```
You compile SkillAlign skill cards for LLM agents. Return
      only valid JSON.
Be faithful to the source and do not invent unavailable
      tools, credentials, APIs, or guarantees.
Create interface views with different cognitive functions,
      not merely different lengths.
Treat checklist as legacy checkpoint material; the main
      procedural interface is workflow.
Prefer precise operational language over generic advice.
When script/config evidence is provided, preserve the
      execution contract.
```

The card-generation input contains the skill id, name, heuristic views, required output schema, interface distinctions, available evidence, and compacted source text. The generator produces fields such as `routing_summary`, `short_hint`, `compressed`, `workflow`, `preconditions`, `failure_modes`, and `keywords`. The quality rules require the model to preserve scripts, arguments, dependencies, inputs, outputs, validation commands, state checks, side effects, and stopping conditions when they appear in the source.

D.2 Agent-Facing Injection

All exposed skill contexts begin with a shared wrapper that instructs the agent to treat skills as optional support rather than hard commands.

```
SkillAlign guidance:
Use the following skill interfaces only when they directly
      help the next action.
Treat them as optional support, not as a command to ignore
      the current observation.
```

The interface-specific body is then appended.

Provider	Benchmark	Backend	Full	Hint	Compressed	Workflow
All	ALFWorld	MiniMax M2.7	47.9 [39.8, 56.1]	68.6 [60.7, 75.8]	**72.1 [64.2, 79.1]**	61.4 [53.2, 69.1]
	ALFWorld	GPT-Codex	88.6 [82.6, 93.3]	92.1 [86.7, 95.8]	**93.6 [88.6, 96.8]**	90.0 [84.3, 94.2]
	ALFWorld	GLM-5	70.7 [62.7, 77.8]	80.7 [73.5, 86.7]	**82.1 [75.0, 87.9]**	76.4 [69.0, 82.7]
	SkillsBench	MiniMax M2.7	14.5 [12.1, 17.1]	16.8 [14.2, 19.6]	**17.2 [14.6, 20.0]**	15.6 [13.1, 18.2]
	SkillsBench	GPT-Codex	24.8 [21.6, 28.2]	29.4 [25.9, 33.0]	**30.8 [27.2, 34.5]**	27.6 [24.2, 31.2]
	SkillsBench	GLM-5	20.4 [17.5, 23.5]	24.0 [20.9, 27.3]	**25.1 [21.9, 28.4]**	22.8 [19.8, 26.0]
Vector-top-k	ALFWorld	MiniMax M2.7	73.6 [66.0, 80.5]	**75.0 [67.4, 81.8]**	69.3 [61.3, 76.6]	64.3 [56.1, 72.0]
	ALFWorld	GPT-Codex	92.9 [87.4, 96.4]	**94.3 [89.2, 97.4]**	94.0 [88.8, 97.2]	91.4 [85.5, 95.4]
	ALFWorld	GLM-5	80.7 [73.5, 86.7]	**83.6 [76.7, 89.2]**	82.9 [75.9, 88.6]	78.6 [71.0, 84.9]
	SkillsBench	MiniMax M2.7	16.4 [13.9, 19.1]	15.6 [13.1, 18.2]	**18.1 [15.4, 21.0]**	16.2 [13.7, 18.9]
	SkillsBench	GPT-Codex	31.8 [28.2, 35.5]	28.9 [25.4, 32.5]	**32.2 [28.6, 35.9]**	29.6 [26.1, 33.2]
	SkillsBench	GLM-5	24.8 [21.7, 28.1]	23.4 [20.3, 26.6]	**26.2 [22.9, 29.6]**	24.4 [21.2, 27.7]
Graph-top-k	ALFWorld	MiniMax M2.7	66.4 [58.3, 73.9]	68.6 [60.7, 75.8]	**71.4 [63.4, 78.6]**	63.6 [55.4, 71.3]
	ALFWorld	GPT-Codex	93.6 [88.6, 96.8]	92.1 [86.7, 95.8]	**95.0 [90.2, 97.8]**	91.4 [85.9, 95.3]
	ALFWorld	GLM-5	80.0 [72.7, 86.1]	76.4 [69.0, 82.7]	**84.3 [77.5, 89.7]**	75.7 [68.2, 82.1]
	SkillsBench	MiniMax M2.7	17.9 [15.2, 20.8]	16.3 [13.8, 19.0]	**19.0 [16.2, 22.0]**	17.4 [14.8, 20.2]
	SkillsBench	GPT-Codex	32.8 [29.2, 36.5]	30.5 [26.9, 34.2]	**34.6 [30.9, 38.4]**	31.5 [27.9, 35.2]
	SkillsBench	GLM-5	25.4 [22.2, 28.8]	23.9 [20.8, 27.2]	**27.1 [23.8, 30.6]**	24.6 [21.4, 27.9]

Table 9: Task-level bootstrap 95% confidence intervals for SkillAlign under the three evaluated candidate providers. Intervals are obtained from 10,000 bootstrap resamples of the evaluation tasks. Bold and underlined values denote the highest and second-highest point estimates, respectively, within each provider–benchmark–backend setting and do not imply statistically significant rankings. The intervals quantify uncertainty under task resampling rather than backend run-to-run stochasticity.

Provider	Benchmark	Backend	Full	Generic Summary	Compressed
Vector-top$k$	ALFWorld	MiniMax M2.7	**73.6**	67.9	69.3
	ALFWorld	GPT-Codex	92.9	93.4	**94.0**
	ALFWorld	GLM-5	80.7	81.8	**82.9**
	SkillsBench	MiniMax M2.7	16.4	17.2	**18.1**
	SkillsBench	GPT-Codex	31.8	31.9	**32.2**
	SkillsBench	GLM-5	24.8	25.4	**26.2**
Graph-top$k$	ALFWorld	MiniMax M2.7	66.4	68.7	**71.4**
	ALFWorld	GPT-Codex	93.6	94.2	**95.0**
	ALFWorld	GLM-5	80.0	82.1	**84.3**
	SkillsBench	MiniMax M2.7	17.9	18.4	**19.0**
	SkillsBench	GPT-Codex	32.8	33.6	**34.6**
	SkillsBench	GLM-5	25.4	26.1	**27.1**

Table 10: Generic length-matched summarization analysis on the full evaluation sets. `Vector-top`k uses $k = 5$ for ALFWorld and $k = 8$ for SkillsBench. Generic Summary uses the same candidate skills and approximately the same rendered context budget as Compressed. Bold and underlined values denote the highest and second-highest point estimates within each row, respectively.

For example, the workflow interface uses ordered steps:

```
=== Skill Workflow: <skill name> ===
<routing summary>
1. <step with check>
2. <step with dependency>
3. <step with stop condition>
```

The compressed interface uses an operational specification:

```
=== Compressed Skill: <skill name> ===
Trigger: ...
Inputs/State: ...
Do: ...
Output: ...
Watch: ...
```

The hint interface contains only a short cue, while the full interface preserves the raw skill document.

D.3 Learned Exposure Policy Prompt

The learned exposure policy predicts an interface, not a skill set. Candidate skills have already been supplied by the provider. A compact prompt skeleton is shown below.

```
Choose the best SkillAlign exposure interface for this task
       and selected candidate skills.
Return only JSON with key interface_mode.
Do not choose which skills to retrieve. The provider has
      already selected the candidates.

Task ID: <task id>
Agent model: <backend model>
Available modes: none, full, hint, compressed, workflow

Task:
<task description>

Initial observation:
<optional observation>

Candidate skill count: <k>
Candidate skill ids: <ids>

Selected candidate skill cards:
<routing summary>
<hint view>
<compressed view>
<workflow view>
<full raw text status>

Decision criterion:
Choose the mode that maximizes expected task success while
      avoiding unnecessary context cost and negative
      transfer.

Return only:
{"interface_mode": "<mode>"}
```

E Exposure Policy Data and Replay Evaluation

E.1 Counterfactual Supervision

We construct exposure-policy data from auxiliary counterfactual runs. For a task and candidate provider, the same candidate skills are evaluated under multiple fixed exposure interfaces. The resulting reward, rendered context cost, and trajectory length are used to derive oracle-style labels and pairwise preferences.

Given a task x and interface m, we compute a utility score:

$$U_x(m) = R_x(m) - \alpha C_x(m) - \beta T_x(m),$$

where $R_x(m)$ is task reward or success, $C_x(m)$ is normalized rendered context cost, and $T_x(m)$ is normalized trajectory length. The highest-utility interface provides an SFT label, while interface pairs with sufficient utility margin provide DPO preference pairs.

E.2 Policy Learning

The policy model receives the task description, provider metadata, interface definitions, and selected skill-card content, and predicts a single interface label for the candidate set. We use a single label per task-provider pair because the fixed-mode counterfactuals observe outcomes at the candidate-set level. This also keeps the policy aligned with the main experimental question: how should the selected skill set be exposed to the agent?

SFT provides a warm start from oracle-style labels, while DPO uses pairwise counterfactual preferences to distinguish cases where one interface is more useful or less costly than another. In this study, we do not use online reinforcement learning because it would require substantially more environment interaction and would confound exposure-policy learning with additional exploration.

E.3 Replay Evaluation

For learned-policy evaluation on ALFWorld, we use off-policy replay. The learned policy predicts an interface for each task and provider, and the evaluator looks up the corresponding fixed-interface trajectory result. This is much cheaper than rerunning every environment episode, but it is reported as replay evaluation rather than fresh online interaction.

Item	Setting
SFT data	Card-aware oracle-style interface labels from counterfactual outcomes
DPO data	Pairwise preferences between exposure interfaces under the same task/provider setting
SFT epochs	5
DPO epochs	2
SFT learning rate	2×10^{-4}
DPO learning rate	5×10^{-6}
DPO preference coefficient	$\beta = 0.1$
Per-device batch size	1
Gradient accumulation steps	16
Precision	bfloat16

Table 11: Training settings for the learned exposure-policy model.

The learned-policy results should therefore be interpreted as a policy-learning analysis rather than as a fully optimized deployment setting. We report both reward and interface-selection distribution, and compare learned policies against oracle exposure to estimate the remaining headroom for adaptive interface selection.

E.4 Training Details

The learned exposure policy is trained separately from the main agent backends. It is used only for the replay-based exposure-selection analysis in Section 4.6. We train LoRA adapters on Qwen3-4B model. The policy receives the task description, provider metadata, candidate skill-card content, and interface definitions, and predicts one exposure interface for the selected candidate skill set. The training parameters are shown in Table 11.

F Computational Budget

The main evaluation cost comes from running fixed-interface counterfactual experiments under multiple candidate providers, exposure interfaces, benchmarks, and agent backends. ALFWorld evaluations are lightweight text-environment runs, while SkillsBench evaluations are more expensive because they require containerized task execution. The learned exposure-policy model is trained with LoRA and is substantially cheaper than full model training. Our training takes 1 NVIDIA A100 GPU for approximately 50 GPU hours in total (including trial experiments).

In our implementation, vector retrieval uses cached embeddings and local NumPy matrix search rather than a separate vector database. SkillsBench experiments require containerized execution, so disk usage and image startup failures are monitored during evaluation.