SKILLDYG: DYNAMIC GROUPING FOR STRUCTURED SKILL RETRIEVAL

Anonymous authors
Paper under double-blind review

ABSTRACT

Skill-augmented agents reuse procedural knowledge from previously solved tasks to improve task execution. For complex tasks, however, ranking skills by semantic similarity alone is insufficient: the usefulness of a skill depends on both its capability and its relations to other skills. Existing methods use structured skill libraries to encode skill roles and relations for retrieval, but often construct the organization only once from skill descriptions, without sufficiently distinguishing when skills apply or how they relate. Errors in that organization can then persist, causing retrieval to confuse candidates that appear relevant to the same task but differ in applicable conditions or overlook redundancy and incompatibility among them. We propose SkillDyG, a framework that dynamically constructs skill groups through validated updates and uses the resulting organization to guide retrieval. In each update round, SkillDyG uses local metadata describing candidate skills' capabilities and execution conditions to propose their placement relative to the current skill groups. It then jointly revises affected groups and reviews the candidate organization before committing any changes. The validated organization provides global metadata that distinguishes each skill's role within its group and captures how that group relates to others. A dual-head retriever combines both views to distinguish similar candidates and retrieve skills that are individually relevant and collectively compatible. Across five backbone models on SkillsBench and ALFWorld, SkillDyG consistently achieves the highest task reward, with an average relative improvement of 11.3% over the strongest baseline on SkillsBench. All related resources and analysis are collected for the community at https://anonymous.4open.science/r/SkilIDyG/.

1 INTRODUCTION

Skill-augmented agents reuse procedural knowledge from previously solved tasks to improve task execution (He et al.; Zheng et al., 2025). As agents gain experience across tasks and environments, they accumulate this knowledge in libraries of reusable skills (Cui et al., 2026; Li et al., 2026b; Zhang et al., 2026). To apply that experience to a new task, an agent retrieves a focused set of skills rather than consulting the entire library (Zhao et al., 2024; Wang et al., 2023). A common approach ranks candidate skills by semantic similarity to the task query (Zheng et al., 2026; Liu et al., 2026c). This ranking can identify plausible candidates, but complex tasks also require information about their roles and dependencies, neither of which can be determined from similarity to the task query alone.

Structured skill libraries seek to provide retrieval with information beyond independent task–skill similarity. Prior work has explored ontologies and capability hierarchies for skill discovery (Liang et al., 2026; Li et al., 2026a). Graph of Skills (GoS) explicitly represents dependencies among individual skills (Liu et al., 2026a), while GoSkills organizes related skills into groups with functional roles (Zeng et al., 2026). Together, these approaches augment retrieval with organizational signals about how skills are related rather than treating each candidate independently (Li et al., 2026c).

However, existing structured skill libraries still face two critical limitations in practice. First, their organization may be too coarse to distinguish candidates that appear relevant to the same task but differ in functional role or applicable conditions (Ding, 2026; Wang et al., 2026a). Figure 1 illustrates this limitation by showing that a fixed structured library can group relevant skills together while still failing to distinguish their roles or capture the dependencies needed for coordinated use. Structured

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

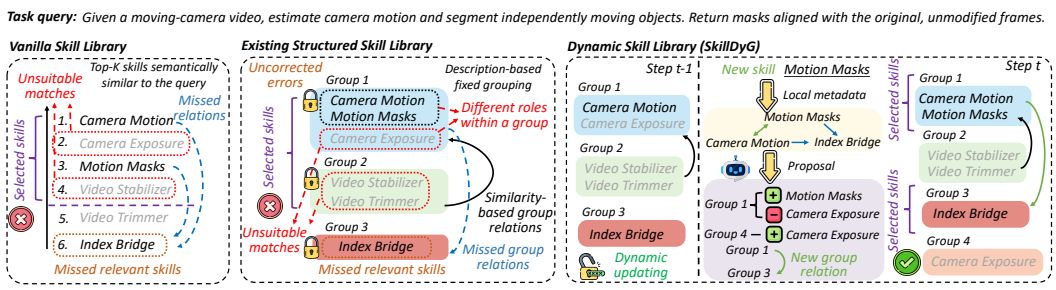


Figure 1: Motivation for dynamically maintained skill organization. Semantic retrieval can select plausible but inapplicable skills and miss relevant dependencies, while fixed structured libraries preserve errors in group boundaries, roles, and relations. SkillDyG instead uses incoming skills to revise affected groups and their relations before validation and commitment, providing reliable organizational evidence for retrieving skills that are individually relevant and collectively compatible.

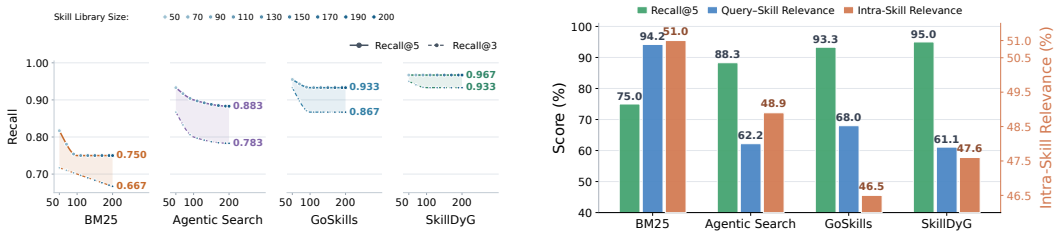
retrieval must therefore determine not only which skills are relevant, but also which are appropriate and how they should work together (Yao et al., 2026; Zhao et al., 2026). Otherwise, selected skills may be redundant or incompatible (Liu et al., 2026b; Chen et al., 2026; Wang et al., 2026b). Second, existing methods often construct the organization once from skill descriptions and keep it fixed (Liu et al., 2026a; Zeng et al., 2026). As the skill library grows, earlier placements, group boundaries, or relations may become inaccurate, allowing organizational errors to persist and mislead retrieval.

These limitations call for a skill organization that remains open to revision as further skills are considered, rather than being fixed after a single construction pass. Constructing such an organization raises three challenges: (i) *Avoiding source-task bias in skill placement.* Skill descriptions often retain the terminology and context of the tasks that produced them. Organizing skills directly from these descriptions can follow incidental context rather than the capabilities and execution conditions needed for reuse. (ii) *Revising affected parts of the existing organization jointly.* A placement that appears reasonable in isolation may change how existing skills are distinguished or how they relate to one another. Its effects must be considered in the context of the current organization, rather than handled as an independent assignment. (iii) *Validating organizational changes before retrieval.* Even a plausible revision can leave conflicting roles or invalid relations elsewhere in the organization. The revised structure must therefore be checked before it becomes evidence for skill selection.

To address these challenges, we propose **SkillDyG**, a framework that dynamically constructs skill groups through validated updates and uses the resulting organization to guide retrieval. In each update round, SkillDyG uses local metadata describing skills' capabilities and execution conditions to propose provisional placements in existing or new groups. It then jointly revises affected groups, including their boundaries, member roles, and relations to other groups, so that each placement is considered in the current organization. The organization is reviewed for consistency, with conflicts returned for revision and only validated changes committed. The validated organization provides global metadata about group membership, member roles, coordination, and relations between groups. At retrieval time, a local head evaluates whether skills fit the task, while a global head uses the organization to distinguish similar candidates and assess how skills can work together. Combining the two views yields skills that are individually relevant and collectively compatible.

Our contributions are as follows:

- We identify two limitations in skill retrieval: distinguishing candidates relevant to the same task and assessing whether selected skills can work together. We show why an organization constructed once from skill descriptions may leave both unresolved.
- We introduce SkillDyG, which dynamically constructs skill groups by jointly revising affected groups and validating organizational changes before commitment. Its dual-head retriever combines local skill metadata with global organizational context for retrieval.
- Across five backbone models on SkillsBench and ALFWorld, SkillDyG achieves the highest task reward, improving over the strongest baseline by 11.3% on average on SkillsBench. The results show that validated group updates better support skill selection and coordination.



(a) Recall as the skill library grows.

(b) Local textual signals versus task-level recall.

Figure 2: Retrieval diagnostics under skill-library growth. Left: Recall is measured on nested SkillsBench libraries of increasing size; circles are interpolated only for visualization. Right: Recall@5 is aggregated over 20 tasks at library size 200, while query to skill relevance and within set skill cohesion are measured on a fixed *mars-clouds-clustering* case rather than across all tasks.

2 Preliminary Study

We examine how retrieval changes as the skill library expands, what retrieval failures arise from fixed grouping, and whether local textual similarity reflects task-level skill coverage. These diagnostics provide direct evidence for the design choices behind SkillDyG.

Retrieval as the library changes. We compare BM25, Agentic Search, GoSkills, and SkillDyG on nested SkillsBench libraries of increasing size, using the same task set at every snapshot. As shown in Figure 2a, Recall declines for the unstructured methods as the candidate set grows. GoSkills maintains higher Recall, showing the benefit of group structure, but still degrades as more skills are introduced. SkillDyG shows the smallest decline across library sizes, suggesting that an organization updated through validated revisions better preserves retrieval coverage as the candidate set changes.

Failures of fixed grouping. We analyze the GoSkills-style fixed organization on SkillsBench and find failures at two stages in Figure 3. During group selection, 15% of tasks miss at least one group containing a relevant skill, showing that the fixed organization can remove needed skills before skill-level ranking begins. Even when the relevant group is selected, irrelevant members can displace relevant skills from the top-k ranking. Notably, 81% of these displacement cases occur in groups with at least 10 skills, showing that most such failures occur in broader groups where more members compete for selection. Together, these results show that fixed grouping can limit retrieval at both stages by restricting which groups are searched and by increasing competition within selected groups.

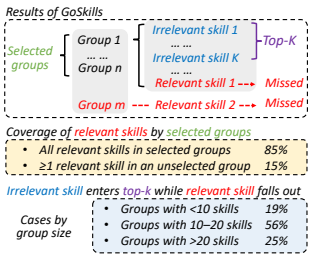


Figure 3: Two retrieval failures under fixed grouping.

Limits of local similarity. We compare aggregate Recall@5 over 20 tasks with local textual diagnostics from a fixed *mars-clouds-clustering* case. As shown in Figure 2b, stronger local textual scores do not necessarily correspond to higher task-level Recall. Because the diagnostics are computed at different aggregation levels, this is a qualitative counterexample rather than a correlation estimate. It shows that textual proximity can identify plausible skills without establishing whether the retrieved set covers required capabilities or whether those skills can work together.

Takeaway. These diagnostics motivate three choices in SkillDyG. Skill placement should rely on capability and execution conditions, affected groups should be revised jointly, and organizational changes should be validated before commitment. Retrieval should combine local evidence with the validated organization to identify skills that are individually relevant and collectively compatible.

3 Methodology

In this section, we present SkillDyG, which dynamically constructs skill groups through validated updates and uses the resulting organization together with local skill metadata to guide task-time

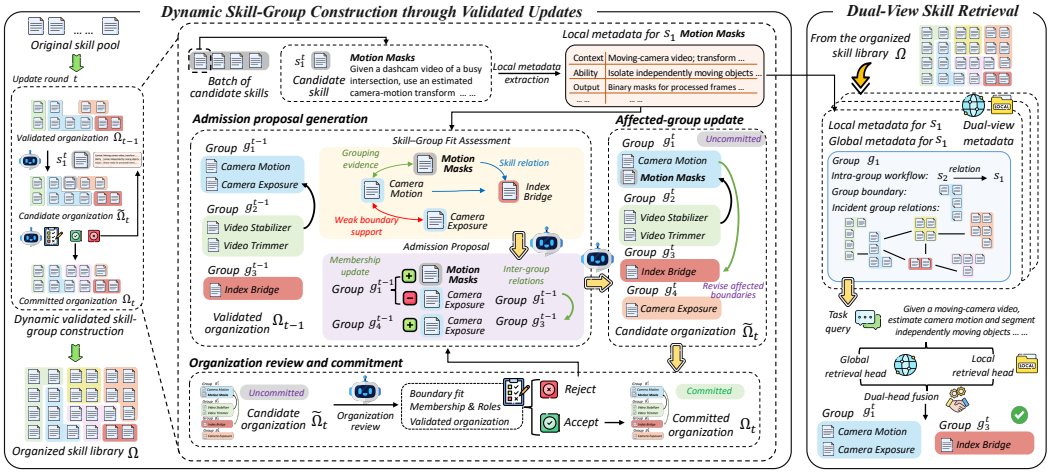


Figure 4: Overview of SkillDyG. Candidate skills are represented by local metadata and receive placement proposals to existing or new groups under the validated organization Ω_{t-1}. Proposals targeting the same group are considered together, and the affected groups are revised and reviewed before Ω_t is committed. The committed organization provides global metadata for dual-head retrieval.

skill retrieval. We first define the skill library and organizational state in Section 3.1, then describe validated skill-group updates in Section 3.2, and finally present dual-head retrieval in Section 3.3.

3.1 Skill Library and Organizational State

At update round t, SkillDyG receives skill entries $\Delta\mathcal{S}_t = \{s_i^t\}_{i=1}^{N_t}$ and starts from the current skill library $\mathcal{S}_{t-1}$ and validated organization Ω_{t-1}. After identifier alignment and near-duplicate resolution, the retained records form the candidate set $\mathcal{C}_t \subseteq \Delta\mathcal{S}_t$, which may contain both new skills and updates to existing ones. For each $s_i \in \mathcal{C}_t$, local metadata m_i^{loc} describes its reusable capability and execution conditions for later group placement decisions. The local metadata for round t is

$$\Delta\mathcal{M}_t^{\mathrm{loc}} = \{m_i^{\mathrm{loc}} \mid s_i \in \mathcal{C}_t\}. \tag{1}$$

The organizational state is $\Omega_t = (\mathcal{G}_t, \mathcal{E}_t^{\mathrm{inter}})$, where $\mathcal{G}_t$ is the set of skill groups and $\mathcal{E}_t^{\mathrm{inter}}$ contains inter-group relations. Each group $g_j^t = (b_j^t, M_j^t, R_j^t, W_j^t)$ specifies its capability boundary b_j^t, members M_j^t, group-specific roles R_j^t, and supported coordination W_j^t. A skill may belong to multiple groups and play a different role in each. The coordination structure W_j^t includes an execution order only when supported by the available evidence for that group.

3.2 Validated Dynamic Skill-Group Construction

SkillDyG updates the organization through placement proposals, group revision, and review. These three stages address the construction challenges introduced earlier. All placement proposals in a round are made against the same validated state Ω_{t-1}, after which the affected groups are revised jointly to form a candidate organization. The three stages are summarized below.

$$\mathcal{P}_t = A_{\mathrm{app}}(\Delta\mathcal{M}_t^{\mathrm{loc}}, \Omega_{t-1}), \qquad \widetilde{\Omega}_t = A_{\mathrm{upd}}(\Omega_{t-1}, \mathcal{P}_t), \qquad \Omega_t = A_{\mathrm{rev}}(\widetilde{\Omega}_t, \Omega_{t-1}), \tag{2}$$

where A_{app} prepares placement proposals $\mathcal{P}_t$, each specifying a target group and its supporting rationale. A_{upd} aggregates proposals targeting the same group to revise the affected groups and form the candidate organization $\widetilde{\Omega}_t$. The candidate organization remains provisional until A_{rev} checks it against Ω_{t-1}, with the validated state fixed throughout the round. If review finds a repairable conflict, its feedback is returned to the placement stage for another revision. Changes that remain invalid are rejected, and only the validated organization is committed as Ω_t.

Capability-Based Placement Proposals. Skill descriptions can contain details from their source tasks that should not determine stable group membership. SkillDyG therefore uses local metadata

focused on reusable capability and execution conditions, including relevant constraints and required tools or artifacts when available. During this stage, the group catalog remains fixed at Ω_{t-1} so that all candidates are evaluated against the same validated organization. The placement agent uses group-level information but does not inspect other members of a target group. A skill may be proposed for several groups, with a separate reason for each proposal. Each proposal is defined as

$$p_{ig} = (s_i, g, \rho_{ig}), \qquad g \in \mathcal{T}_i \subseteq \mathcal{G}_{t-1} \cup \{g_{\text{new}}\}, \tag{3}$$

where $\mathcal{T}_i$ is the set of target groups, g_{new} denotes a request to form a new group when no existing group provides a suitable fit, and ρ_{ig} explains the capability- and condition-based fit. A candidate may therefore have several proposals in one round. The proposal set is given by

$$\mathcal{P}_t = \{p_{ig} \mid s_i \in \mathcal{C}_t,\ g \in \mathcal{T}_i\}. \tag{4}$$

All proposals remain provisional until the affected groups have been revised and the resulting organization has been reviewed, and no change is committed at this stage.

Joint Revision of Affected Groups. The update stage collects all proposals for each target group and considers them together with its current members, since several plausible placements can jointly change the group's boundary and how its members should be distinguished. For an existing group g_j^{t-1}, the proposals targeting that group are defined as

$$\mathcal{P}_{t,j} = \{p_{ig} \in \mathcal{P}_t \mid g = g_j^{t-1}\}. \tag{5}$$

The corresponding candidate revision is

$$\widetilde{g}_j^t = U_{\text{grp}}(g_j^{t-1}, \mathcal{P}_{t,j}). \tag{6}$$

The revision may update the group boundary, candidate and existing memberships, member roles, and supported coordination. It considers these parts together so that a new placement does not create an unclear boundary, conflicting roles, or unsupported coordination among members. An existing member may also be removed or reassigned, in which case the destination group is treated as affected. Proposals for g_{new} are used to form candidate new groups when no existing group provides a suitable fit. The revised groups and changes to their inter-group relations together form the candidate organization $\widetilde{\Omega}_t$ in Eq. 2, while unaffected groups keep their previous state. This limits revision to the affected part of the organization rather than rebuilding all groups after each update.

Organization Review and State Commitment. The review stage compares $\widetilde{\Omega}_t$ with the validated state Ω_{t-1}. It checks new applications and relevant existing members against the candidate group boundaries and execution conditions, and also checks whether roles, inter-group relations, and coordination remain consistent across the affected parts of the organization. When an ordered workflow lacks enough support, only the supported coordination is retained. When validation identifies an inconsistency, it returns feedback on the corresponding application, group boundary, role, or relation. The affected application or group is then revised, producing a new candidate organization for validation. This review and repair loop continues until the current batch passes validation or the remaining invalid changes are rejected. Validation therefore determines which proposed group boundaries, memberships, roles, and relations are recorded in Ω_t. Library updates and duplicate-resolution details are provided in Appendix A. Records aligned to existing skills may instead update their local metadata. Only the committed organization Ω_t provides global evidence to retrieval and becomes the reference for the next round. Here, *validated* means that an update passes the stated consistency checks, not that the resulting organization is uniquely optimal.

3.3 Dual-Head Retrieval with Local and Global Views

Local and Global Metadata. SkillDyG uses local skill metadata together with global metadata derived from the committed organization to support task-time skill retrieval. The two views capture individual skill fit and organizational context, respectively. We define the two metadata views as

$$\mathcal{M}_t^{\text{loc}} = \{m_i^{\text{loc}} \mid s_i \in \mathcal{S}_t\}, \qquad \mathcal{M}_t^{\text{glob}} = F_{\text{glob}}(\Omega_t). \tag{7}$$

The local view records each skill's reusable capability and execution conditions, including relevant tools, artifacts, and constraints when available. The global view records validated group boundaries and memberships, each skill's role relative to other members, supported coordination, and inter-group relations. Thus, local metadata describes a skill on its own, while global metadata describes how it is organized with other skills. This separation lets retrieval use both individual fit and organizational context. It helps distinguish similar skills and assess whether they can work together.

Table 1: **End-to-end task performance and execution cost.** We report task success rate (Reward, %), cumulative input tokens, and execution steps across five backbones on SkillsBench and ALFWorld. Ret. denotes task-specific skill retrieval, and Coord. denotes explicit inter-skill execution relationships. Bold and underline mark **the best** and second-best results. Green marks relative improvements in Reward and Tokens over the second-best.

Backbone	Method	Skill Access Ret.	Skill Access Coord.	SkillsBench Reward	SkillsBench Tokens	SkillsBench Steps	ALFWorld Reward	ALFWorld Tokens	ALFWorld Steps
Gemini-3.1 Pro	Vanilla Skills	×	×	38.5	854,965	26.2	95.1	94,112	17.3
	Vector Skills	✓	×	40.2	622,717	22.6	95.3	56,744	16.2
	GoS	✓	✓	45.6	601,531	19.8	96.7	59,186	14.9
	GoSkills	✓	✓	50.4	616,969	18.5	98.2	54,604	14.6
	SkillDyG	✓	✓	**57.3** (+13.7%)	**570,379** (+5.2%)	**17.1**	**99.3** (+1.1%)	**50,578** (+7.4%)	**13.7**
GPT-5.4	Vanilla Skills	×	×	29.8	860,464	17.1	97.2	108,502	17.7
	Vector Skills	✓	×	26.4	580,447	10.4	97.1	77,218	19.0
	GoS	✓	✓	36.8	530,074	9.6	97.9	62,896.7	15.9
	GoSkills	✓	✓	49.7	693,041	10.1	97.5	66,536	16.1
	SkillDyG	✓	✓	**52.4** (+5.4%)	**509,112** (+4.0%)	**9.2**	**98.9** (+1.0%)	**52,648** (+16.3%)	**13.9**
Claude-4.6	Vanilla Skills	×	×	30.0	865,963	21.5	94.2	121,394	18.4
	Vector Skills	✓	×	27.6	606,672	**10.9**	95.5	73,490	18.7
	GoS	✓	✓	39.1	476,512	11.2	95.7	98,456	15.7
	GoSkills	✓	✓	47.5	727,876	14.8	97.9	119,338	15.6
	SkillDyG	✓	✓	**54.4** (+14.5%)	**410,041** (+13.9%)	13.5	**99.3** (+1.4%)	**57,462** (+21.8%)	**14.5**
Qwen3.5	Vanilla Skills	×	×	21.1	873,295	28.8	92.0	103,528	21.8
	Vector Skills	✓	×	18.4	723,112	24.7	93.9	79,796	19.2
	GoS	✓	✓	27.9	**540,014** (+1.0%)	**18.4**	94.2	75,556	16.7
	GoSkills	✓	✓	33.9	752,340	24.9	94.3	65,202	16.4
	SkillDyG	✓	✓	**38.6** (+13.9%)	545,729	24.1	**97.1** (+3.0%)	**56,946** (+12.7%)	**14.8**
Kimi-k2.5	Vanilla Skills	×	×	26.1	882,459	24.5	90.1	108,940	19.5
	Vector Skills	✓	×	22.8	650,861	20.1	90.5	69,392.5	19.5
	GoS	✓	✓	34.3	**451,993** (+8.9%)	20.6	94.3	**52,402.5** (+0.8%)	17.2
	GoSkills	✓	✓	42.5	715,248	18.9	93.6	63,346.3	16.9
	SkillDyG	✓	✓	**46.3** (+8.9%)	496,031	**17.7**	**96.4** (+2.2%)	52,798.8	**15.1**

Dual-Head Selection. For a task query q, the local head assesses which individual skills meet the required capabilities and execution conditions. The global head uses group boundaries and member roles to distinguish similar candidates and uses supported relations to assess complementary, redundant, or conflicting combinations. Their query-conditioned outputs are defined as

$$H_q^{\mathrm{loc}} = H_{\mathrm{loc}}\big(q, \mathcal{M}_t^{\mathrm{loc}}\big), \qquad H_q^{\mathrm{glob}} = H_{\mathrm{glob}}\Big(q, \mathcal{M}_t^{\mathrm{glob}}\Big). \tag{8}$$

Fusion combines the two outputs to select a task-dependent skill set. Along with the selected skills, it provides the validated role, relation, and coordination context needed by the executor to organize their use during task execution. The final retrieval output is

$$Y_q = (\mathcal{S}_q^\star, \Gamma_q) = \Psi\big(H_q^{\mathrm{loc}}, H_q^{\mathrm{glob}}\big), \tag{9}$$

where $\mathcal{S}_q^\star \subseteq \mathcal{S}_t$ is the selected skill set and Γ_q contains the roles and coordination context supported by Ω_t. Fusion favors complementary task coverage while avoiding unsupported, redundant, or conflicting combinations, yielding skills that are individually relevant and collectively compatible.

4 Experiments

In this section, we conduct comprehensive experiments, aiming to answer the following questions: **RQ1 (Effectiveness).** How effectively does SkillDyG support task execution through multi-skill retrieval, compared with existing skill-retrieval methods? **RQ2 (Ablation).** How do the key components and parameter choices of SkillDyG contribute to retrieval quality and organization quality? **RQ3 (Expansion).** As the skill library grows, can SkillDyG preserve retrieval quality for previously supported tasks and incorporate newly available relevant skills into its retrieval results? Note that, for a comprehensive evaluation, we provide additional experiments on execution efficiency, backbone sensitivity, parameter sensitivity, case studies, etc., presented in Appendices F–I.

4.1 EXPERIMENTAL SETTINGS

Datasets. We evaluate end-to-end task execution on two agent benchmarks: SkillsBench (Li et al., 2026b) and ALFWorld (Shridhar et al., 2021). SkillsBench uses 95 containerized tasks to evaluate skill-assisted task completion, whereas ALFWorld uses 274 episodes to evaluate interactive household-task execution. Detailed descriptions and evaluation scales are provided in Appendix D.1.

Baselines. Our end-to-end baselines follow the GoSkills comparison setting (Zeng et al., 2026) and represent different skill-access and organization strategies. We adopt Vanilla Skills and Vector Skills as the direct-library-access and flat semantic-retrieval controls defined in Graph of Skills (Liu et al., 2026a). GoS uses dependency-based organization (Liu et al., 2026a), whereas GoSkills uses functional-role-based group retrieval (Zeng et al., 2026). For the retrieval-quality analysis, we additionally include SkillRouter (Zheng et al., 2026), Skill-Usage (Liu et al., 2026c), and SkillDAG (Bai et al., 2026). We further compare these baseline methods on a 1,000-skill candidate pool from SkillRet (Cho et al., 2026) to assess retrieval over a large-scale open skill library. Detailed baseline descriptions are provided in Appendix D.2.

Evaluation Metrics. For end-to-end execution, we report strict task success rate (Reward), mean cumulative input tokens (Tokens), and mean execution steps (Steps). For retrieval quality, we use predefined task–skill relevance annotations and report macro-averaged Precision@k for $k \in \{3, 5\}$. Formal metric definitions, counting scopes, and failure-handling rules are provided in Appendix D.3.

Implementation Details. All experiments are conducted on a server equipped with NVIDIA A800 GPUs. End-to-end evaluations cover Gemini-3.1 Pro (Google DeepMind, 2026), GPT-5.4 (OpenAI, 2026), Claude-4.6 (Anthropic, 2026), Qwen3.5 (Qwen Team, 2026), and Kimi-k2.5 (Kimi Team et al., 2026).

Table 2: **Retrieval performance (%).** We evaluate macro-averaged Precision@k among the top-k returned skills on SkillsBench and ALFWorld.

	SkillsBench		ALFWorld	
Method	P@3	P@5	P@3	P@5
Unstructured retrieval				
Vanilla Skills	43.9	29.7	40.8	38.8
Vector Skills	58.9	42.1	40.5	38.9
SkillRouter	55.1	40.8	18.5	18.5
Skill-Usage	<u>64.2</u>	44.2	<u>70.2</u>	52.9
Structured retrieval				
GoS	60.4	42.7	38.6	42.7
SkillDAG	56.5	40.8	35.3	30.3
GoSkills	59.6	<u>44.6</u>	56.2	<u>53.8</u>
SkillDyG	**68.4** (+6.5%)	**48.8** (+9.4%)	**73.0** (+4.0%)	**68.6** (+27.5%)

4.2 TASK PERFORMANCE AND RETRIEVAL QUALITY (RQ1)

To address RQ1, we evaluate SkillDyG from three perspectives: end-to-end task execution, execution cost, and standalone skill-retrieval quality. Table 1 compares end-to-end task performance, input-token consumption, and execution steps across five backbone models on SkillsBench and ALFWorld. Table 2 directly evaluates the quality of returned skill lists using fixed task–skill relevance annotations. The main results in Tables 1 and 2 lead to the following observations.

Obs. 1. SkillDyG consistently improves end-to-end task performance across backbones. Table 1 shows that SkillDyG achieves the highest task reward on SkillsBench and ALFWorld with all five backbones. On SkillsBench, it raises the average reward from 44.8% with GoSkills to 49.8%; on ALFWorld, it further improves reward by 1.1–2.8 percentage points over GoSkills despite its already strong performance. By jointly revising affected groups and validating member relations and coordination structure, SkillDyG keeps the organizational evidence used during retrieval updated as the skill library expands, enabling further end-to-end performance gains.

Obs. 2. SkillDyG achieves higher task performance with a more compact execution context. Table 1 reports mean cumulative input tokens and mean execution steps per task. By defining clear group boundaries, distinguishing the roles of group members, and providing validated collaborative relationships, SkillDyG achieves the highest task rewards. For example, with GPT-5.4 on ALFWorld, SkillDyG raises reward from 97.5% to 98.9%, while reducing mean input tokens from 66,536 to 52,648 and mean execution steps from 16.1 to 13.9. The simultaneous reduction in cumulative input

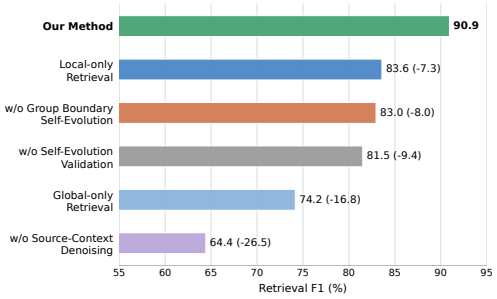


Figure 5: **Ablation study.** Retrieval F_1 (%) across component ablations.

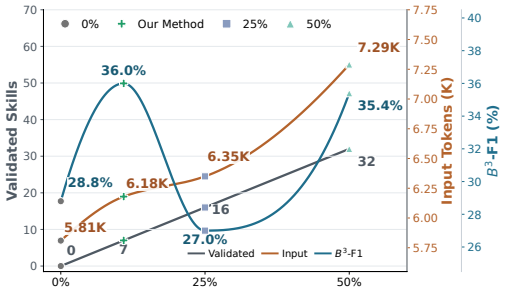


Figure 6: **Validation sampling analysis.** Validated skills, input tokens, and B^3-F_1.

tokens and execution steps indicates that this context enables the agent to follow execution logic more clearly, thereby avoiding ineffective or redundant attempts during downstream task execution.

Obs. 3. SkillDyG achieves superior retrieval quality with a dynamically maintained skill organization. SkillDyG achieves the best P@3 and P@5 on both benchmarks: 68.4 and 48.8 on SkillsBench, improving over the strongest baselines by 6.5% and 9.4%, respectively, and 73.0 and 68.6 on ALFWorld, improving by 4.0% and 27.5%. These gains further indicate that semantically similar skills need not share the same applicable conditions or functional roles, and that combining two views better matches skills to task requirements when complementary skills must work together.

4.3 Ablation and Parameter Analysis (RQ2)

To address RQ2, we examine the contributions of individual components and the effects of parameter settings. Figures 5–7 summarize the main results, while additional results are provided in Appendix G.

Obs. 4. Every component of SkillDyG contributes to retrieval quality. As shown in Figure 5, removing source-context denoising causes the largest F_1 drop, by 26.5 percentage points, indicating that organizing skills directly from source-task terminology and context can make placement susceptible to task-specific details. Removing group-boundary self-evolution or self-evolution validation lowers F_1 by 8.0 and 9.4 points, respectively, showing that affected groups and their relations must be considered and updated as a whole, then validated before serving as retrieval evidence. Using only the local or global view lowers F_1 by 7.3 and 16.8 points, respectively, showing that the two sources of information cannot substitute for one another: local metadata provides the capability- and condition-level basis for matching candidate skills, whereas global metadata from the validated organization further determines which locally relevant candidates are appropriate and how they should work together, including their distinct roles, valid coordination relations, and the execution order.

Obs. 5. Adaptive settings efficiently balance maintenance cost and retrieval quality. Figure 6 shows that, at a common maintenance update, adaptive validation checks only seven existing skills and obtains 36.0% B^3-F_1 with 6.18K input tokens. In contrast, fixed 50% validation checks 32 skills, consumes 7.29K tokens, and obtains 35.4% B^3-F_1. This result shows that organization quality can be maintained without scaling the validation scope linearly with group size. Figure 7 further shows that fixed retrieval budgets that are too small miss complementary skills, whereas large budgets introduce irrelevant or easily confused candidates. Adaptive dual-view retrieval selects 3.17 skills on average and achieves 77.2% F_1, outperforming the best fixed dual-view setting by 10.1 percentage points. Together, these results show that adaptive settings control validation cost during maintenance and balance skill coverage against retrieval noise according to task needs. Specifically, adaptive sampling prioritizes newer group members, while adaptive retrieval adjusts the number of selected skills to task requirements. The two choices reduce unnecessary checks and limit extra candidates that may increase retrieval noise.

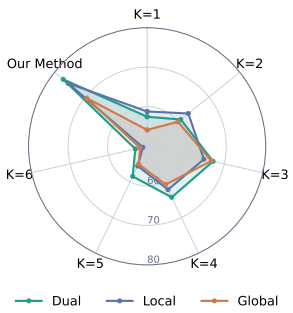
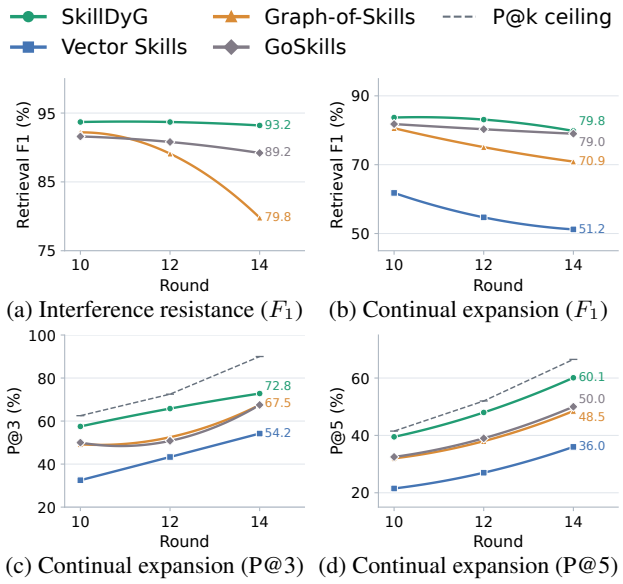


Figure 7: **Retrieval size analysis.** Retrieval F_1 (%) with fixed and adaptive selection. Spokes denote fixed or adaptive selection; curves compare views.

4.4 Continual Skill-Library Expansion (RQ3)

To address RQ3, we study whether SkillDyG remains effective under continual skill-library expansion. Figure 8(a) evaluates whether retrieval on a fixed existing-task cohort resists interference from newly added irrelevant candidates, whereas Figure 8(b–d) evaluates whether retrieval on a designated new-skill task cohort incorporates newly available relevant skills promptly. Full evaluation protocols and results are provided in Appendix H.



(a) Interference resistance (F_1) (b) Continual expansion (F_1)

(c) Continual expansion (P@3) (d) Continual expansion (P@5)

Figure 8: **Skill-library expansion analysis.** We report retrieval F_1 on an existing-task cohort to assess robustness against irrelevant skill additions (a), and retrieval after continual admission of available skills with F_1, P@3, and P@5 (b–d). Dashed lines in (c–d) mark the attainable $P@k$ ceilings at each expansion round, given the gold skills available in the library for the designated new-skill tasks.

Obs. 6. SkillDyG resists retrieval interference from irrelevant skill additions. In Figure 8(a), we evaluate a fixed existing-task cohort whose gold skills are already available. All subsequently added skills are therefore irrelevant candidates. As the library grows from 128 to 219 skills, SkillDyG's retrieval F_1 changes only from 93.7% to 93.2%, whereas GoSkills declines from 91.6% to 89.2% and GoS from 92.2% to 79.8%. This result indicates that continuously maintaining and validating group boundaries, member roles, and relations reduces interference from newly added irrelevant skills.

Obs. 7. SkillDyG promptly incorporates newly available relevant skills. In Figure 8(b–d), methods are compared within each checkpoint on a designated new-skill task cohort because the set of available gold skills changes as the library expands. At round 12, SkillDyG reaches 65.8% P@3 and 48.0% P@5, exceeding GoSkills by 15.0 and 9.0 points and GoS by 13.3 and 10.0 points, respectively. Because all methods use the same library at each checkpoint, these gaps reflect retrieval differences under equal skill availability rather than differences in the available gold-skill set. This result indicates that the dynamic organization helps newly available relevant skills align with existing group boundaries, roles, and relations, so that they enter the ranked results promptly even as the skill library continues to expand and introduces more candidate skills over successive update rounds.

5 Conclusion

Skill-augmented agents benefit from reusable skills, but semantic similarity alone cannot determine how skills should be organized and combined for complex reasoning tasks that require the coordinated use of multiple capabilities. Graph-based and group-based methods expose useful dependencies and roles, yet fixed offline grouping cannot reconsider errors in group membership, role assignment, or boundaries when the library grows and the relationships among its skills change. SkillDyG addresses these limitations by constructing the organization through successive validated updates. It derives capability-focused local metadata for admission proposals, jointly updates affected groups under the current organizational state, and reviews each candidate organization against the previously validated state before commitment to ensure that group boundaries and member relations remain consistent as the library changes. The resulting global metadata is combined with local metadata by a dual-head retriever to select skills that are individually relevant and collectively compatible. Experiments on SkillsBench and ALFWorld with five backbone models show that SkillDyG attains the strongest task reward and improves retrieval and execution efficiency over existing methods. When the SkillsBench library grows, SkillDyG also better preserves retrieval quality on previously supported tasks and more effectively incorporates newly available relevant skills into its retrieval results.

AI USE STATEMENT

In this work, we used generative AI tools to assist with manuscript writing and polishing. We have not used generative AI tools for other required disclosure tasks, including implementing methods, designing experiments, cleaning datasets, proposing or refining hypotheses, and interpreting results. We have reviewed all AI-assisted work: we manually verified the accuracy and fidelity of generated content and checked the polished text for meaning preservation. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work does not involve human participants, personal data, or human-subject experiments. The evaluated agents operate on benchmark tasks and a controlled skill library. As with other systems that reuse language-model-generated skills, downstream beha vior can reflect errors or biases in the underlying skills and models; validated group updates reduce organizational inconsistencies but do not remove these risks.

REPRODUCIBILITY STATEMENT

We use the public SkillsBench, ALFWorld, and SkillRet benchmarks. The paper and appendices specify the library construction settings, update protocol, retrieval procedure, evaluation metrics, backbone models, prompts, and ablation configurations. We will release the implementation, configuration files, and experiment scripts to support reproduction of the reported results.

REFERENCES

Anthropic. Introducing claude sonnet 4.6, February 2026. URL https://www.anthropic.com/news/claude-sonnet-4-6.

Tong Bai, Zhenglin Wan, Pengfei Zhou, Xingrui Yu, Yang You, and Ivor W Tsang. Skilldag: Self-evolving typed skill graphs for llm skill selection at scale. *arXiv preprint arXiv:2606.03056*, 2026.

Yanping Chen, Weijie Shi, Wen Yang, and Jiajie Xu. Task decomposition-guided reranking for adaptive agent skill retrieval. *arXiv preprint arXiv:2607.06283*, 2026. doi: 10.48550/arXiv.2607.06283. URL https://arxiv.org/abs/2607.06283.

Hongcheol Cho, Ryangkyung Kang, and Youngeun Kim. SkillRet: A large-scale benchmark for skill retrieval in LLM agents. *arXiv preprint arXiv:2605.05726*, 2026. doi: 10.48550/arXiv.2605.05726. URL https://arxiv.org/abs/2605.05726.

Zicai Cui, Zihan Guo, Weiwen Liu, and Weinan Zhang. Skillpager: Query-adaptive intra-skill navigation via semantic node retrieval. *arXiv preprint arXiv:2606.00822*, 2026.

Jiandong Ding. Skillresolve-bench: Measuring and resolving same-capability ambiguity in agent skill retrieval, June 2026. URL https://arxiv.org/abs/2606.10388.

Google DeepMind. Gemini 3.1 pro model card, February 2026. URL https://deepmind.google/models/model-cards/gemini-3-1-pro/.

Haixing He, Hanqi Li, Jiexi Wu, Haoran Luo, Shuai Zhao, Kai Yu, and Xiaobao Wu. Agent skills: A comprehensive survey of construction, runtime, evolution, evaluation, and governance.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, S. H. Cai, Yuan Cao, et al. Kimi k2.5: Visual agentic intelligence. *arXiv preprint arXiv:2602.02276*, 2026. doi: 10.48550/arXiv.2602.02276. URL https://arxiv.org/abs/2602.02276.

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

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. Api-bank: A comprehensive benchmark for tool-augmented llms. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 3102–3116, 2023.

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

Xiaoyuan Li, Moxin Li, Keqin Bao, Yubo Ma, Wenjie Wang, Dayiheng Liu, and Fuli Feng. Skillgraph: Skill-augmented reinforcement learning for agents via evolving skill graphs. *arXiv preprint arXiv:2605.12039*, 2026c.

Yuan Liang, Ruobin Zhong, Haoming Xu, Chen Jiang, Yi Zhong, Runnan Fang, Jia-Chen Gu, Shumin Deng, Yunzhi Yao, Mengru Wang, et al. Skillnet: Create, evaluate, and connect ai skills. *arXiv preprint arXiv:2603.04448*, 2026.

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

Shuo Liu, Yutong Yang, Haohao Xiao, Mouxing Yang, and Xi Peng. Imagine before retrieval: Prospective skill retrieval for LLM agents. *arXiv preprint arXiv:2609.01642*, 2026b. doi: 10.48550/arXiv.2609.01642. URL https://arxiv.org/abs/2609.01642.

Xukun Liu, Zhiyuan Peng, Xiaoyuan Yi, Xing Xie, Lirong Xiang, Yuchen Liu, and Dongkuan Xu. Toolnet: Connecting large language models with massive tools via tool graph. *arXiv preprint arXiv:2403.00839*, 2024.

Yujian Liu, Jiabao Ji, Li An, Tommi Jaakkola, Yang Zhang, and Shiyu Chang. How well do agentic skills work in the wild: Benchmarking LLM skill usage in realistic settings. *arXiv preprint arXiv:2604.04323*, 2026c. doi: 10.48550/arXiv.2604.04323. URL https://arxiv.org/abs/2604.04323.

OpenAI. Introducing GPT-5.4, March 2026. URL https://openai.com/index/introducing-gpt-5-4/.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive apis. In *Advances in Neural Information Processing Systems*, volume 37, pp. 126544–126565, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/e4c61f578ff07830f5c37378dd3ecb0d-Paper-Conference.pdf.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=dHng2O0Jjr.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, volume 36, pp. 68539–68551, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/d842425e4bf79ba039352da0f658a906-Paper-Conference.pdf.

Zhengliang Shi, Yuhan Wang, Lingyong Yan, Pengjie Ren, Shuaiqiang Wang, Dawei Yin, and Zhaochun Ren. Retrieval models aren't tool-savvy: Benchmarking tool retrieval for large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 24497–24524. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-acl.1258. URL https://aclanthology.org/2025.findings-acl.1258/.

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

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Cote, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=0IOX0YcCdTn.

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

Tianle Wang, Yanghe Zou, Xiang Liu, Ziyao Huang, Chenchen Fu, and Weiwei Wu. Beyond task-only matching: Personalized skill routing with counterfactual evaluation, August 2026a. URL https://arxiv.org/abs/2608.28241.

Zifei Wang, Wei Wen, Qiang Ji, Ruizhi Qiao, and Xing Sun. Skill is not document: A query-conditional benchmark and two-stage retriever for LLM agent skill routing. *arXiv preprint arXiv:2606.03565*, 2026b. doi: 10.48550/arXiv.2606.03565. URL https://arxiv.org/abs/2606.03565.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Yue Yao, Shengyuan Wang, Xin Chen, Minke Zhang, Jia He, Bingjun Luo, and Tom Gedeon. Skill-trace: Traversing a query-skill graph for composable llm agents. *arXiv preprint arXiv:2608.02356*, 2026.

Kun Zeng, Yu Huo, Siyu Zhang, Zi Ye, Yuecheng Zhuo, Haoyue Liu, Yuquan Lu, Junhao Wen, and Xiaoying Tang. Group of skills: Group-structured skill retrieval for agent skill libraries. *arXiv preprint arXiv:2605.06978*, 2026.

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

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

Xinyu Zhao, Zhen Tan, Vaishnav Tadiparthi, Nakul Agarwal, Kwonjoon Lee, Ehsan Moradi Pari, Hossein Nourkhiz Mahjoub, and Tianlong Chen. Generative skill composition for llm agents. *arXiv preprint arXiv:2606.32025*, 2026.

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

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

Appendix

CONTENTS

1 Introduction **1**

2 Preliminary Study **3**

3 Methodology **3**

3.1 Skill Library and Organizational State 4

3.2 Validated Dynamic Skill-Group Construction 4

3.3 Dual-Head Retrieval with Local and Global Views 5

4 Experiments **6**

4.1 Experimental Settings 7

4.2 Task Performance and Retrieval Quality (RQ1) 7

4.3 Ablation and Parameter Analysis (RQ2) 8

4.4 Continual Skill-Library Expansion (RQ3) 9

5 Conclusion **9**

A Frequently Asked Questions (FAQs) **15**

A.1 How are group boundaries updated for an incoming batch? 15

A.2 How Does SkillDyG Ensure Reasonable Boundary Evolution? 15

A.3 What Distinguishes Dynamic Organization from Fixed Grouping? 15

A.4 How Do Experiments Substantiate the Advantages of SkillDyG? 16

B Related Work **16**

B.1 Tool Use with LLM Agents 16

B.2 Skill-augmented LLM agents 17

C Details of the Proposed Method **17**

D Experimental Settings and Evaluation Protocols **19**

D.1 Benchmarks 19

D.2 Baselines 19

D.3 Evaluation Metrics 20

D.4 Experimental Configuration 21

E Implementation Details **21**

E.1 Skill Representations 21

E.2 Dynamic Group Maintenance 21

E.3 Adaptive Existing-member Sampling 22

E.4 Dual-Head Retrieval 22

F Additional Results for RQ1 **22**

F.1 Execution Efficiency and Joint Performance 22

F.2 Retrieval with Qwen3.5 24

F.3 Retrieval in SkillRet 24

G Additional Ablation Studies (RQ2) **25**

G.1 Existing-skill Validation Policies 25

H Additional Model Studies(RQ3) **27**

H.1 Retrieval Trajectories under Skill-Library Expansion 27

I Prompt Templates and Case Study **28**

I.1 Prompt Templates 28

I.2 Illustrative Boundary-Repair Case 33

I.3 Error Case 33

J The Use of Large Language Models (LLMs) **35**

J.1 Writing and Formatting Assistance 35

A Frequently Asked Questions (FAQs)

A.1 How are group boundaries updated for an incoming batch?

SkillDyG updates the library and its organization in alternating batchwise phases. At round t, the raw arrivals $\Delta\mathcal{S}_t$ first undergo normalization, identifier alignment, and initial duplicate resolution, yielding the candidate set $\mathcal{C}_t$. Each candidate receives local metadata describing its reusable capability and execution conditions. The candidates are evaluated against the frozen, previously committed organization Ω_{t-1}: the application agent proposes target groups for each candidate with supporting rationales, and the group-boundary update agent aggregates proposals targeting the same group to determine memberships, roles, relations, and the candidate organization $\widetilde{\Omega}_t$ for the current batch.

If validation resolves a record as a canonical duplicate, it is included in $\mathcal{D}_t \subseteq \mathcal{C}_t$ and removed as a separate library entry. The batchwise library update is $\mathcal{S}_t = (\mathcal{S}_{t-1} \cup \mathcal{C}_t) \setminus \mathcal{D}_t$. Validation determines the group boundaries, memberships, roles, and relations recorded in Ω_t.

Validation checks every applicant from the current batch against the updated boundary and samples existing members from each affected group to verify that the revised boundary still supports them. If it identifies an inconsistency in an application, boundary, role, or relation, it provides a diagnosis to the corresponding agent. The agent repairs the application or affected group, reconstructs $\widetilde{\Omega}_t$, and invokes validation again. If no existing group can support a retained skill, the update agent creates a new group for it. Thus, a batch is committed as Ω_t only after this review cycle is complete.

Maintenance and retrieval are phase-separated: SkillDyG completes application, affected-group revision, validation, and repair for the current batch before processing $\Delta\mathcal{S}_{t+1}$ or serving queries, ensuring that each group-boundary evolution is evaluated against a stable committed state and completed before it influences subsequent updates or retrieval. At query time, the local head uses metadata from the completed skill library $\mathcal{S}_t$, whereas the global head uses only the committed organization Ω_t; retrieval therefore never observes a partially updated organization.

A.2 How Does SkillDyG Ensure Reasonable Boundary Evolution?

SkillDyG ensures reasonable boundary evolution through joint validation of new-member fit, existing-member stability, and the evidence supporting roles and relations. For an incoming batch, the application agent prepares placement proposals with target groups and supporting rationales, using candidate-local metadata and the frozen prior organization. The group-boundary update agent then aggregates proposals targeting the same group to determine candidate memberships, roles, relations, and group boundaries.

After the candidate organization is constructed, the validation agent assesses whether the proposed state transition satisfies the organizational consistency conditions. It uses the frozen prior organization, candidate organization, structured application evidence, and local metadata of the relevant skills. The validation agent does not reuse upstream agents' hidden reasoning traces or conversational states, nor does it generate new membership applications or boundary revisions. For each affected group, it checks every new applicant against the updated boundary and verifies whether the proposed roles and relations are supported by the skill's capabilities, artifacts, tools, constraints, and available checks. It also adaptively audits existing members to identify mismatches introduced by the boundary update. When an unsupported membership, boundary drift, role inconsistency, or invalid relation is identified, the issue is routed to the corresponding application, boundary, role, or relation component for repair. The repaired candidate organization must pass validation again before it is committed as the formal organization supporting subsequent retrieval and future batch updates.

A.3 What Distinguishes Dynamic Organization from Fixed Grouping?

Similarity-based retrieval can identify individually plausible skills, but cannot determine their applicable conditions, group-specific roles, or whether they can coordinate with one another. Fixed group structures provide organizational context, but cannot revise group boundaries, member roles, or inter-group relations when an incoming skill exposes a mismatch. SkillDyG proposes placements based on reusable capabilities and execution conditions, jointly revises the affected groups instead of admitting an incoming skill in isolation, and validates the resulting organization before it informs

retrieval. The resulting global metadata complements local skill evidence and distinguishes locally similar skills through validated roles and supported relations. Consistent with this mechanism, removing source-context denoising, group-boundary evolution, or validation reduces retrieval F_1 in our ablations. The continual-expansion results further show reduced interference from irrelevant additions and more prompt incorporation of newly available relevant skills as the library expands.

A.4 How Do Experiments Substantiate the Advantages of SkillDyG?

Similarity-based retrieval alone can identify skills that appear individually plausible, but it cannot determine their applicable conditions, group-specific roles, or whether they can coordinate with one another. A fixed organization provides some structural context, yet cannot revise established boundaries, member roles, or relations when incoming skills reveal a mismatch. SkillDyG evaluates these limitations through complementary experiments.

First, RQ1 and the efficiency analyses in the appendix examine whether organizational updates introduce unacceptable execution overhead. Across five backbone models on SkillsBench and ALFWorld, the paper reports end-to-end task reward, retrieval precision, input-token consumption, and execution steps. The appendix further reports output tokens, wall-clock time, joint performance profiles, alternative-backbone retrieval results, and results on the open-library SkillRet setting. For maintenance, the paper also compares the token cost and organizational quality of different validation-sampling policies. These results show that dynamic maintenance does not merely trade additional cost for better performance, but yields an observable balance among retrieval quality, execution outcomes, and maintenance overhead. Second, RQ3 directly examines the limitations of fixed organization as the library changes. The experiments fix cohorts of existing tasks and newly available-skill tasks, then reevaluate them at four stages as the library expands from 128 to 219 skills. One set of experiments tests whether irrelevant incoming skills interfere with retrieval for existing tasks, while the other tests whether newly available relevant skills can be incorporated promptly. The continual-expansion results therefore evaluate not only final performance, but also the stability and adaptability of dynamic maintenance throughout organizational change. Finally, RQ2 tests whether the individual design components are necessary. Removing source-context denoising, group-boundary evolution, or evolution validation reduces retrieval F_1 by 26.5, 8.0, and 9.4 percentage points, respectively. Retaining only the local or global view reduces F_1 by 7.3 and 16.8 points. These results correspondingly support the roles of capability- and condition-based representations, joint updates of affected groups, post-update validation, and the local and global combination.

Together, these complementary evaluations assess SkillDyG from end-to-end effectiveness, maintenance efficiency, continual robustness, and component necessity. They provide converging empirical evidence that dynamically maintaining and validating the organization improves skill retrieval and execution while avoiding full existing-member validation at each maintenance update.

B Related Work

B.1 Tool Use with LLM Agents

Language-model agents can extend their capabilities by combining reasoning with external actions. ReAct interleaves reasoning traces with actions, while Toolformer learns when and how to invoke a small set of APIs (Yao et al., 2023; Schick et al., 2023). As the number of available tools grows, the central problem shifts from deciding whether to call a tool to finding the right tools in a large collection. API-Bank, Gorilla, and ToolLLM develop benchmarks and methods for learning to use broad API collections (Li et al., 2023; Patil et al., 2024; Qin et al., 2024). ToolRet further shows that retrievers that perform well on conventional information-retrieval benchmarks can still struggle with tool retrieval, and that retrieval errors reduce downstream task success (Shi et al., 2025). Beyond ranking tools independently, ToolNet represents relations among tools as a graph to support access to large tool collections (Liu et al., 2024). These studies establish the importance of scalable discovery and structured access to external functions. SkillDyG instead focuses on reusable procedural skills: it asks how groups of related skills should be maintained as the library changes, and how the maintained organization should guide multi-skill retrieval.

B.2 SKILL-AUGMENTED LLM AGENTS

A complementary line of work studies how agents acquire reusable knowledge from experience. Reflexion converts feedback into verbal reinforcement, ExpeL extracts reusable insights from prior tasks, and Voyager builds an expanding library of executable skills in an embodied environment (Shinn et al., 2023; Zhao et al., 2024; Wang et al., 2023). SkillWeaver similarly discovers and improves reusable skills for web agents (Zheng et al., 2025). These methods primarily address how skills are learned or refined. SkillDyG takes the arrival of new skills as its starting point and studies how they should be incorporated without degrading the organization of the existing library.

Recent work also studies skill libraries at a larger scale. SkillsBench provides paired evaluations of agent performance with and without curated skills and shows that focused skill sets can be more effective than exhaustive ones (Li et al., 2026b). SkillNet organizes skills with a shared ontology and relational connections, while AgentSkillOS uses a capability hierarchy for skill discovery and composes selected skills into execution pipelines (Liang et al., 2026; Li et al., 2026a). Most closely related to our work, GoS constructs a dependency graph over a skill library and retrieves bounded bundles that include required dependencies (Liu et al., 2026a). GoSkills organizes skills into role-aware groups and retrieves compact execution contexts in which the roles of selected skills are made explicit (Zeng et al., 2026). Both methods demonstrate the value of structure for multi-skill retrieval, but their graph or group structure is constructed offline for a given library. SkillDyG addresses the complementary problem of maintaining this structure as new skills are added. It uses new skills to propose revisions to affected groups, validates the revised organization before committing it, and exposes the resulting local and global information to retrieval.

C DETAILS OF THE PROPOSED METHOD

At round t, SkillDyG first normalizes, aligns, and initially deduplicates the arriving records $\Delta\mathcal{S}_t$ to obtain $\mathcal{C}_t$. Each candidate derives local metadata and is processed as part of one batch against the frozen organization Ω_{t-1}. The application agent prepares $\mathcal{P}_t$ by proposing target groups with supporting rationales; the update agent then aggregates proposals by target group and determines the corresponding memberships, roles, relations, and group revisions. If validation later resolves a candidate as a canonical duplicate, that record is removed as a separate entry from the library; all other candidates remain in $\mathcal{S}_t$. The batch alternates between application repair, affected-group revision, and validation until its organizational update passes validation and is committed.

Algorithm 1 makes the completed library and committed organization available to retrieval. Algorithm 2 specifies maintenance across skill batches and the subsequent dual-head retrieval step, while Algorithm 3 details one maintenance round. Together, the algorithms ensure that local retrieval is constructed from the completed library and global retrieval from the committed organization.

Algorithm 1: Dual-head Skill Retrieval

Input: Task query q; completed skill library $\mathcal{S}_t$; validated state Ω_t
Output: Executor-facing skill selection Y_q

Construct $\mathcal{M}_t^{\mathrm{loc}} = \{m_i^{\mathrm{loc}} \mid s_i \in \mathcal{S}_t\}$ and $\mathcal{M}_t^{\mathrm{glob}} = F_{\mathrm{glob}}(\Omega_t)$ after the current batch has completed maintenance;
Compute $H_q^{\mathrm{loc}} \leftarrow H_{\mathrm{loc}}(q, \mathcal{M}_t^{\mathrm{loc}})$ to assess individual skill fit;
Compute $H_q^{\mathrm{glob}} \leftarrow H_{\mathrm{glob}}(q, \mathcal{M}_t^{\mathrm{glob}})$ to assess organizational context;
$Y_q \leftarrow \Psi(H_q^{\mathrm{loc}}, H_q^{\mathrm{glob}})$; `// Fuse both heads for task-dependent selection`
return Y_q *with the associated roles and coordination context*;

Algorithm 2: Overall SkillDyG Procedure

Input: Candidate skill batches $\Delta\mathcal{S}_1, \ldots, \Delta\mathcal{S}_T$; initial skill library $\mathcal{S}_0$; initial validated state Ω_0; task query q
Output: Skill library $\mathcal{S}_T$; validated organizational trajectory $\Omega_{1:T}$; executor-facing selection Y_q

for $t \leftarrow 1$ **to** T **do**
  Normalize $\Delta\mathcal{S}_t$, align identifiers, and remove near duplicates; let $\mathcal{C}_t$ be the remaining update set;
  Derive candidate-local metadata and initialize $\mathcal{S}_t \leftarrow \mathcal{S}_{t-1} \cup \mathcal{C}_t$;
  Generate placement proposals and supporting rationales $\mathcal{P}_t$ from the local metadata of $\mathcal{C}_t$ and Ω_{t-1};
  Construct $\widetilde{\Omega}_t \leftarrow A_{\mathrm{upd}}(\Omega_{t-1}, \mathcal{P}_t)$ by jointly updating affected groups and incident relations;
  Validate every new applicant and an adaptive sample of existing members, obtaining validation feedback;
  while *validation feedback identifies a duplicate or an inconsistency* **do**
    Remove any records resolved as canonical duplicates from $\mathcal{S}_t$ and $\mathcal{C}_t$;
    Diagnose and repair the affected application, boundary, role, or relation;
    Regenerate the affected applications and rationales from the repaired state;
    Construct $\widetilde{\Omega}_t \leftarrow A_{\mathrm{upd}}(\Omega_{t-1}, \mathcal{P}_t)$ by jointly updating affected groups and incident relations;
    Validate every new applicant and an adaptive sample of existing members, obtaining validation feedback;
  Commit the validated candidate organization $\Omega_t \leftarrow \widetilde{\Omega}_t$;
Derive $\mathcal{M}_T^{\mathrm{loc}} = \{m_i^{\mathrm{loc}} \mid s_i \in \mathcal{S}_T\}$ and $\mathcal{M}_T^{\mathrm{glob}} = F_{\mathrm{glob}}(\Omega_T)$, then compute $H_q^{\mathrm{loc}} = H_{\mathrm{loc}}(q, \mathcal{M}_T^{\mathrm{loc}})$ and $H_q^{\mathrm{glob}} = H_{\mathrm{glob}}(q, \mathcal{M}_T^{\mathrm{glob}})$;
$Y_q \leftarrow \Psi(H_q^{\mathrm{loc}}, H_q^{\mathrm{glob}})$; `// Fuse both heads for task-dependent selection`
return $\mathcal{S}_T, \Omega_{1:T}, Y_q$ *with the corresponding executor-facing roles and coordination context*;

Algorithm 3: Validated Dynamic Group Maintenance at Round t

Input: Candidate batch $\Delta\mathcal{S}_t$; skill library $\mathcal{S}_{t-1}$; previous validated state Ω_{t-1}
Output: Updated skill library $\mathcal{S}_t$; committed state Ω_t

Normalize and align $\Delta\mathcal{S}_t$ to obtain $\mathcal{C}_t$; derive candidate-local metadata and set $\mathcal{S}_t \leftarrow \mathcal{S}_{t-1} \cup \mathcal{C}_t$;
Generate placement proposals and supporting rationales $\mathcal{P}_t$ from candidate-local metadata and the frozen state Ω_{t-1};
Construct $\widetilde{\Omega}_t \leftarrow A_{\mathrm{upd}}(\Omega_{t-1}, \mathcal{P}_t)$ by jointly revising affected groups and incident relations;
foreach *affected group* g **do**
  Validate every new member assigned to g;
  Let M_g be the existing members before the update and sample m_g members without replacement according to the adaptive weights;
  Validate the sampled existing members against the candidate boundary, roles, and workflow;
while *validation feedback identifies a duplicate or an inconsistency* **do**
  Remove any records resolved as canonical duplicates from $\mathcal{S}_t$ and $\mathcal{C}_t$;
  Diagnose and repair the affected application, boundary, role, or relation;
  Regenerate the affected applications and rationales;
  Construct $\widetilde{\Omega}_t \leftarrow A_{\mathrm{upd}}(\Omega_{t-1}, \mathcal{P}_t)$ by jointly revising affected groups and incident relations;
  foreach *affected group* g **do**
    Validate every new member assigned to g;
    Let M_g be the existing members before the update and sample m_g members without replacement according to the adaptive weights;
    Validate the sampled existing members against the candidate boundary, roles, and workflow;
Commit $\Omega_t \leftarrow \widetilde{\Omega}_t$;
return $\mathcal{S}_t, \Omega_t$;

D EXPERIMENTAL SETTINGS AND EVALUATION PROTOCOLS

D.1 BENCHMARKS

We evaluate SkillDyG in three complementary settings. SkillsBench and ALFWorld support end-to-end task-execution evaluation and standalone skill-retrieval analysis, allowing us to examine whether retrieval benefits downstream performance. SkillRet instead evaluates retrieval over an open skill library, complementing the interactive task benchmarks with a broader skill source for retrieval. Table 3 summarizes the original purpose and coverage of each benchmark, together with its role.

SkillsBench. We use 95 containerized tasks derived from SkillsBench to evaluate skill-assisted multi-step task execution and retrieval quality. SkillsBench consists of realistic workflows that require domain-specific procedural knowledge. Each task provides an isolated execution environment, task-specific resources, and a skill bundle, while a deterministic verifier determines task completion. Representative workflows include software engineering, cybersecurity, scientific data analysis, financial analysis, media and content processing, and industrial systems. For retrieval evaluation, we score returned skills against predefined task–skill relevance annotations (Li et al., 2026b).

ALFWorld. We evaluate 274 episodes from ALFWorld's text-based `AlfredTWEnv` environment, including 140 *valid seen* and 134 *valid unseen* episodes. The evaluation covers all six household-task types: looking at an object in light, simple pick-and-place, pick-two-objects-and-place, pick-clean-then-place, pick-cool-then-place, and pick-heat-then-place. Agents complete multi-step goals through language observations and actions. Alongside end-to-end task success, we score returned skill lists against predefined task–skill relevance annotations (Shridhar et al., 2021).

SkillRet. We further use SkillRet to evaluate retrieval over an open skill library. SkillRet pairs natural-language user requests with public agent skills and provides binary relevance annotations. Its skills cover software engineering, AI agents, data and machine learning, content creation, business and planning, and information retrieval. Unlike the interactive benchmarks above, SkillRet focuses on identifying appropriate skills from an open collection for each request (Cho et al., 2026).

D.2 BASELINES

Following GoS (Liu et al., 2026a), we group the compared methods by whether they use explicit inter-skill relations to organize retrieval. Here, unstructured refers to the absence of organization across skills, not the format of individual skill documents. Vanilla Skills and Vector Skills serve as direct-access and flat-retrieval references, respectively, throughout the comparative evaluation.

Unstructured Skill Access and Retrieval. Vanilla Skills gives the agent direct access to the available skill library without a separate retriever that narrows candidates in advance. The agent must inspect, select, and coordinate skills from the exposed library. It is thus a direct-access control: no dependency graph or role-based organization guides selection or execution within a task.

Vector Skills represents each skill independently in an embedding space and retrieves top-k candidates by task similarity. It returns a flat ranking of individually relevant skills without dependency expansion or functional grouping. This control measures the value added by explicit inter-skill organization under the same task-query and candidate-library setting in each matched evaluation.

SkillRouter uses a two-stage retrieve-and-rerank pipeline for skill selection from large libraries (Zheng et al., 2026). A bi-encoder first retrieves a candidate shortlist, and a cross-encoder jointly processes the query and each candidate's full content to refine the ranking. Both stages use skill names, descriptions, and body text, allowing the ranking to consider procedural details beyond short metadata. The method improves individual ranking but supplies neither dependency-aware bundles nor role-labeled groups to guide downstream multi-step execution for the agent's tasks.

Skill-Usage evaluates skill use in realistic settings where an agent must retrieve and select skills from a large open collection (Liu et al., 2026c). We use its retrieval setting as a realistic-library reference without an explicit graph or group organization among the skills returned to the agent.

Table 3: Benchmark coverage and roles in our evaluation. The table summarizes each benchmark's original purpose and task coverage, then specifies how it is used in this paper.

Dataset	Original benchmark purpose and coverage	Use in this paper
SkillsBench	Evaluates whether reusable agent skills improve realistic multi-step task completion. The original suite spans 11 domains: software engineering, office and white-collar work, natural science, media and content production, cybersecurity, finance, robotics, manufacturing, energy, mathematics, and healthcare. Tasks use curated skills and deterministic verifiers.	We use 95 containerized tasks for end-to-end execution and fixed-gold retrieval evaluation, with 252 gold-skill instances (2.65 reference skills per task).
ALFWorld	Aligns text-based TextWorld environments with embodied ALFRED worlds for household instruction following. It covers six task types: look-at-object-in-light, pick-and-place-simple, pick-two-objects-and-place, pick-clean-then-place, pick-cool-then-place, and pick-heat-then-place.	We use 274 text-based episodes (140 valid-seen and 134 valid-unseen) to evaluate interactive task success and the relevance of retrieved skills.
SkillRet	Benchmarks retrieval of public agent skills for natural-language user requests in a large open library. Its two-level taxonomy has six major categories—software engineering, AI agents, data and machine learning, content creation, business and planning, and information retrieval—and 18 subcategories.	We use the filtered evaluation split: 6,006 held-out skills, 4,392 queries, and 7,187 binary relevance judgments, to evaluate open-library skill retrieval.

Structured Skill Retrieval. **Graph of Skills (GoS)** constructs a directed multi-relational graph whose nodes represent skills and whose edges encode prerequisite and workflow relations (Liu et al., 2026a). Retrieval starts from semantic and lexical seeds, then propagates relevance through the graph using reverse-aware Personalized PageRank. This process can recover supporting prerequisites that are not strong direct matches to the query. The selected skills are assembled under a context budget into a dependency-aware bundle, making relational coverage part of the bounded retrieval context that informs subsequent agent execution across the selected skill bundle.

GoSkills organizes skills into anchor-centered groups built over a typed skill graph and expands support groups through a group graph (Zeng et al., 2026). It selects a group plan and compresses it into a bounded set of atomic skill payloads. The resulting context includes an execution contract with Start, Support, Check, and Avoid fields that clarify the entry point, supporting capabilities, visible requirements, and cautions. This baseline makes the intended use of retrieved skills explicit through role-aware group organization during task execution in the designated environment.

SkillDAG represents inter-skill relations with a typed directed graph and exposes it as an agent-callable retrieval interface (Bai et al., 2026). Each search returns vector matches, typed graph neighbors, and conflict signals as separate sources of evidence for skill selection. Its propose-then-commit protocol lets the agent register execution-backed graph edges, so the relation graph accumulates information across episodes. This update incrementally adds relations rather than jointly revising and validating group boundaries, member roles, and group-level coordination for downstream retrieval.

D.3 Evaluation Metrics

We evaluate methods at both the end-to-end task-execution and skill-retrieval levels. For task execution, **Reward** denotes the strict task success rate: a SkillsBench run succeeds only when its container verifier returns *reward = 1.0*, whereas an ALFWorld episode succeeds only when the environment emits its success signal. **Tokens** in the main table is computed by summing provider-reported input tokens across execution calls within each task and then averaging over tasks, including prompts and conversation history resubmitted at each call. ALFWorld token accounting excludes the generation of precomputed retrieval artifacts.

For retrieval evaluation, let $\mathcal{G}(q)$ denote the set of relevant skills for a query q, and let $R_k(q)$ denote the top-k skills returned by a method. We report macro-averaged Precision@k:

$$\mathrm{P@}k = \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} \frac{|R_k(q) \cap \mathcal{G}(q)|}{k},$$

where $k \in \{3, 5\}$. A returned skill is counted as relevant only when its canonical skill identity matches a relevance annotation for the corresponding task. This metric measures the fraction of retrieved skills that are relevant under a limited skill budget. For SkillsBench, we exclude infrastructure or harness failures only when they occur before the agent begins execution, and report their number and causes separately in the appendix. Tool errors, timeouts, environment-interaction failures, and incomplete tasks occurring after agent execution begins are counted as task failures.

Execution steps. On SkillsBench, **Steps** denotes the mean number of executor turns per task. Each executor response counts as one turn; tool calls and their returned observations within that turn do not add separate steps. Initial task prompts, verifier calls, and API retries are not counted separately. On ALFWorld, **Steps** denotes the mean number of actions recorded per episode, including environment-interaction actions, invalid or repeated actions, and `think:` turns recorded as actions. It therefore describes the length of the agent's execution process rather than only actions that successfully change the environment state. For both benchmarks, we average steps over all of the evaluated tasks.

Output tokens and execution time. Output tokens are accumulated across execution-stage model calls within each task and then averaged over tasks. Time denotes the mean wall-clock execution time per ALFWorld episode in the corresponding execution environment, measured in seconds.

D.4 Experimental Configuration

All experiments use a server equipped with NVIDIA A800 GPUs. End-to-end evaluations cover Gemini-3.1 Pro, GPT-5.4, Claude-4.6, Qwen3.5, and Kimi-k2.5. For retrieval-quality comparisons, we use GPT-5.4 and Qwen3.5 as backbone models across all reported retrieval evaluations.

E Implementation Details

E.1 Skill Representations

After each maintenance round, the system stores local skill information, group-construction records and rationales for that round, and updated global information. The skill library retains every canonical, nonduplicate record from the completed batch. Local retrieval cards are materialized from this library after duplicate resolution; global information records the validated skill-group memberships, member roles, group boundaries, workflow relations, and inter-group relations committed in Ω_t.

E.2 Dynamic Group Maintenance

For dynamic skill-group maintenance, we process arriving skills in batches of 16. At the beginning of each batch, the active group catalog is frozen so that all incoming skills are evaluated against the same group boundaries and member structure; group applications within a batch are processed jointly after the batch closes. Source-task information is used only as a provisional seed during first-round cold start and is excluded from subsequent stable group boundaries, membership assignments, and retrieval indices.

The group-boundary agent produces structured JSON decisions through an OpenAI-compatible API. For each candidate group update, the validation module examines every newly proposed member and audits m_g existing formal members, where M_g denotes the set of existing formal members before the update and m_g is defined below. When validation identifies an inconsistency, it diagnoses whether the application, boundary, role, or relation requires repair and returns the affected component to the corresponding agent. The repaired candidate organization is then revalidated before the batch is committed. Only validated boundaries, member roles, workflow relations, and their associated group updates are committed in Ω_t.

State trace. For every candidate, the maintenance log records its identifier, duplicate-resolution outcome, proposed target groups, validation outcome, repair route, and final group and role. This trace makes the batchwise transition from Ω_{t-1} to Ω_t auditable.

E.3 Adaptive Existing-member Sampling

For a candidate group update, all new applicants are validated. Let M_g be the set of existing formal members before the update. SkillDyG samples

$$m_g = \begin{cases} 0, & |M_g| = 0, \\ \lceil 0.1|M_g| \rceil, & |M_g| > 0 \end{cases}$$

old members without replacement. For member i, let τ_i denote the stored formal-member tenure and let $\Delta_i = \max(0, t - t_i^{\text{last}})$ denote the elapsed maintenance epochs since its last validation. The sampling weight is

$$w_i = 0.05 + \frac{1}{1+\tau_i} + 0.18\Delta_i.$$

At each draw, a member is sampled from the remaining pool R with probability

$$p_i(R) = \frac{w_i}{\sum_{j \in R} w_j}.$$

The selected member is removed from R before the next draw. This rule prioritizes newer members and members overdue for revalidation. Adaptivity concerns selection probabilities; the nominal sample-size ratio remains 10%. Uniform baselines use the same without-replacement procedure with equal weights and ratios $\rho \in \{0, 0.1, 0.25, 0.5, 1\}$; zero explicitly disables old-member checks. The uniform 10% baseline therefore matches the adaptive policy's sample count on the same candidate update. Actual member counts are reported because integer rounding can make distinct nominal ratios select the same number of members in small groups. Repeated validation during repair is included in cost accounting.

E.4 Dual-Head Retrieval

Before these records are used for dual-head retrieval, the system performs consistency checks. Member roles and membership scopes must be valid; no skill may hold primary membership in multiple groups; workflow edges must connect valid members; inter-group relations must connect existing groups; and every formal member must have corresponding local skill information. Only local and global information that passes these checks is used for retrieval.

Table 4: Mean ALFWorld execution time across backbones. The second-shortest times are underlined and the **shortest times** are in bold. Relative time reductions from the second-shortest times are shown in green parentheses.

Method	Gemini-3.1 Pro	GPT-5.4	Claude-4.6	Qwen3.5	Kimi-k2.5
Vanilla Skills	654.9	179.1	214.8	431.6	174.6
Vector Skills	321.7	191.0	84.5	257.8	104.0
GoS	481.6	153.6	100.5	**217.6** (+5.0%)	**73.8** (+9.2%)
GoSkills	375.6	159.9	102.8	253.9	148.8
SkillDyG	**276.5** (+14.1%)	**138.5** (+9.8%)	**82.8** (+2.0%)	229.0	81.3

F Additional Results for RQ1

This section supplements Section 4.2 with execution-efficiency results and joint performance profiles, followed by retrieval results with Qwen3.5 and on the open skill library in SkillRet.

F.1 Execution Efficiency and Joint Performance

ALFWorld execution time. Table 4 compares mean task execution time in seconds across five backbone models on ALFWorld. SkillDyG achieves the shortest execution time with Gemini-3.1 Pro, GPT-5.4, and Claude-4.6, reducing time by 14.1%, 9.8%, and 2.0% relative to the second-best method, respectively; it ranks second with Qwen3.5 and Kimi-k2.5. Compared with GoSkills, SkillDyG achieves higher task reward with shorter execution time across all five backbones.

SkillDyG continually maintains group boundaries, member responsibilities, and workflows, and combines local execution information with global organizational information through dual-head retrieval to select task-relevant skills and their coordination context. Clearer execution relationships can help the agent coordinate multiple skills and may reduce unproductive attempts, providing a possible explanation for the observed time reductions. With Qwen3.5 and Kimi-k2.5, however, SkillDyG takes fewer execution steps but more time than GoS, indicating that overall execution time also depends on model response times and runtime conditions.

GPT-5.4 output tokens. Figure 9 reports mean cumulative output token consumption for GPT-5.4 on ALFWorld. Output tokens are summed across execution-stage model calls within each task and then averaged over 274 tasks. SkillDyG uses 603.1 output tokens per task, reducing consumption by 33.1%, 7.1%, 7.8%, and 7.3% compared with Vanilla Skills, Vector Skills, GoS, and GoSkills, respectively, while achieving the highest task reward and the fewest execution steps.

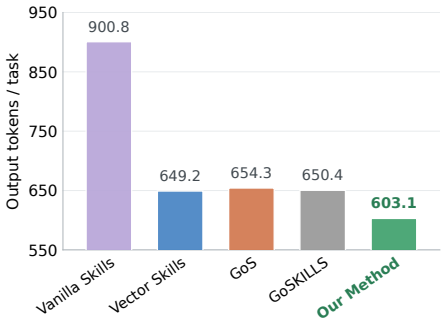


Figure 9: GPT-5.4 output tokens per task on ALFWorld.

During retrieval, SkillDyG combines execution information about individual skills with the maintained group structure, providing task-relevant skills together with their responsibilities and coordination relationships. This context helps identify applicable skills and guides their sequencing and execution, potentially reducing repeated exploration, consistent with the lower step counts and cumulative output consumption. Taken together, the task reward, execution-step, and output-token results show that SkillDyG improves task performance while reducing execution-stage generation overhead.

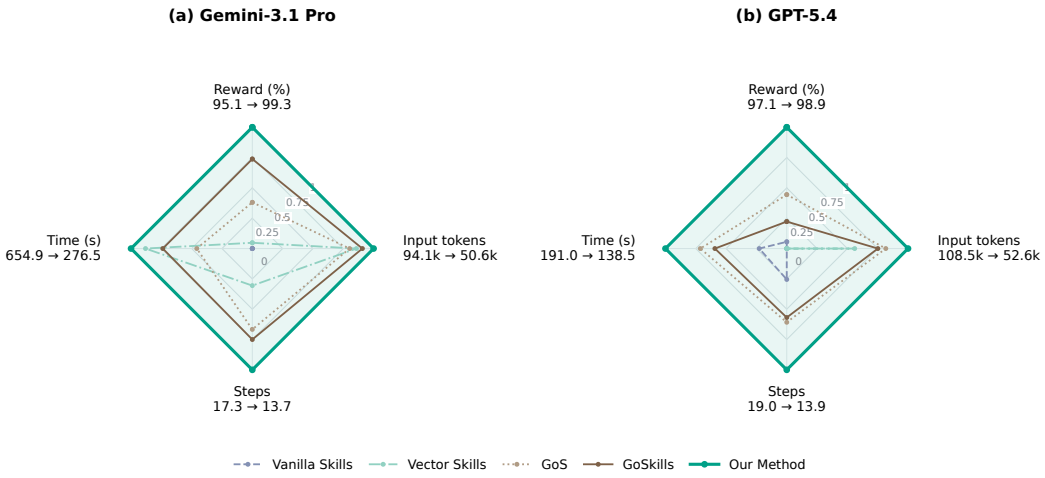


Figure 10: ALFWorld performance profiles for (a) Gemini-3.1 Pro and (b) GPT-5.4.

Joint performance profiles. Figure 10 compares task reward, input token consumption, execution steps, and execution time on ALFWorld for Gemini-3.1 Pro and GPT-5.4. Each metric is min–max normalized across the five methods within each backbone: Reward uses $(x-\min x)/(\max x-\min x)$, while input tokens, Steps, and Time use $(\max x - x)/(\max x - \min x)$. Larger scores indicate better performance; 0 and 1 denote the worst and best reported values, respectively. Axis labels give the original worst-to-best ranges, with token ranges rounded in thousands, so the profiles show relative positions rather than absolute performance ratios.

SkillDyG achieves the best reported value on all four metrics for both backbones, combining higher task reward with lower input consumption, fewer execution steps, and shorter execution time. The baselines exhibit different strengths. With Gemini-3.1 Pro, GoSkills ranks second in reward, input

tokens, and steps, whereas Vector Skills has the second-shortest execution time. With GPT-5.4, GoS ranks second on all four metrics. SkillDyG continually maintains skill-group boundaries, responsibilities, and workflows, and combines local execution information with global organizational information to select a coordination context that provides clearer guidance for skill selection and execution. The resulting performance profiles show improvements in task performance together with reduced execution overhead.

F.2 Retrieval with Qwen3.5

Retrieval with Qwen3.5. Table 5 reports P@3 and P@5 with an alternative backbone to examine the sensitivity of LLM-based skill selection to model choice. Vector Skills and SkillRouter do not query the chat backbone, so their results are unchanged from the GPT-5.4 setting. Other methods vary across backbones and benchmarks; for example, Vanilla Skills achieves 34.6% P@5 on ALFWorld with Qwen3.5, compared with 38.8% with GPT-5.4. This provides a simple robustness check without changing the retrieval setup.

Table 5: **Retrieval precision with Qwen3.5.** P@k is the number of gold skills among the top-k retrieved skills divided by k, macro-averaged over queries.

	SkillsBench		ALFWorld	
Method	P@3	P@5	P@3	P@5
Unstructured retrieval				
Vanilla Skills	<u>61.9</u>	22.7	36.2	34.6
Vector Skills	58.9	42.1	40.5	38.9
SkillRouter	55.1	40.8	18.5	18.5
Skill-Usage	60.4	42.5	<u>56.0</u>	43.0
Structured retrieval				
GoS	60.5	42.7	38.7	43.2
SkillDAG	56.5	40.8	35.3	30.3
GoSkills	57.2	<u>43.6</u>	46.7	<u>48.1</u>
SkillDyG	**65.4** (+5.7%)	**47.2** (+8.3%)	**61.6** (+10.0%)	**53.8** (+11.9%)

Three rows match the GPT-5.4 results exactly, by construction rather than by coincidence. Vector Skills and SkillRouter retrieve with their own embedding and reranking models and never query the chat backbone. SkillDAG does query it, but only to build its typed graph offline: its search returns the ten cosine matches before any graph neighbour, so the graph cannot affect the first five ranks. The two backbones do build different graphs and the retrieved lists diverge beyond rank ten, which P@3 and P@5 never reach.

F.3 Retrieval in SkillRet

Retrieval performance on SkillRet. We further evaluate relevant-skill retrieval in an open library on the SkillRet candidate pool. Unlike SkillsBench and ALFWorld, SkillRet does not provide predefined skill groups, member roles, or workflow relations; this setting therefore directly tests whether SkillDyG can construct an organization suitable for retrieval using only information from the candidate skills. Specifically, SkillDyG initializes its organizational state from the candidate pool and processes skills as continually arriving records. For each candidate, it proposes membership using its local capability description, input–output artifacts, tool cues, and constraints; when no existing group satisfies the boundary conditions, it creates a candidate group. It then jointly revises the affected groups' boundaries, member roles, and relations, and commits the update through the validation–repair loop. Throughout construction, SkillRet query–skill relevance annotations are not used for skill grouping, boundary updates, validation, or retrieval; they are used only to compute the final retrieval metrics.

Table 6: **Retrieval precision on SkillRet.** P@k is the number of gold skills among the top-k retrieved skills divided by k, macro-averaged over queries.

	GPT-5.4		Qwen3.5	
Method	P@3	P@5	P@3	P@5
Unstructured retrieval				
Vanilla Skills	25.4	23.6	<u>40.2</u>	<u>25.2</u>
Vector Skills	33.3	21.3	33.3	21.3
SkillRouter	26.3	17.6	26.3	17.6
Skill-Usage	<u>43.0</u>	<u>26.5</u>	36.7	22.6
Structured retrieval				
GoS	33.9	21.9	33.8	21.9
SkillDAG	32.9	21.6	32.9	21.6
GoSkills	25.2	16.7	19.2	13.1
SkillDyG	**44.5** (+3.5%)	**32.1** (+21.1%)	**43.8** (+9.0%)	**29.7** (+17.9%)

Table 6 reports results on the same candidate pool and evaluation queries. SkillDyG achieves the highest P@3 and P@5 under both GPT-5.4 and Qwen3.5. With GPT-5.4, SkillDyG reaches

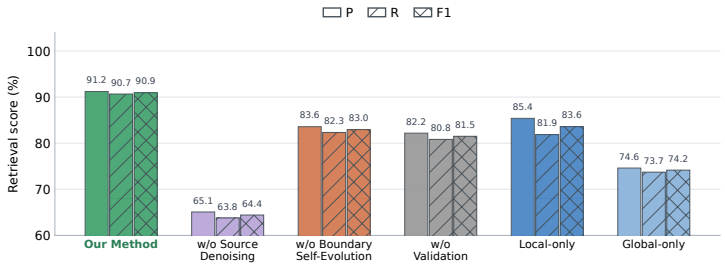
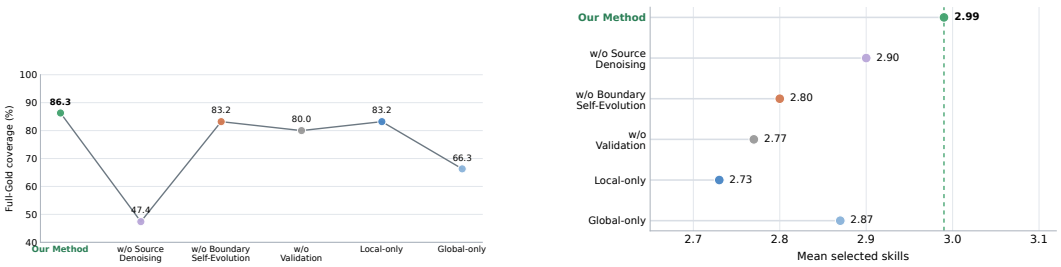


Figure 11: Precision, recall, and F1 for each component-ablation setting. Color identifies the method; bar texture identifies the metric.



(a) Gold-skill coverage on SkillsBench.

(b) Average selected skills on SkillsBench.

Figure 12: Component ablations on SkillsBench.

44.5% P@3 and 32.1% P@5, exceeding the strongest baseline, Skill-Usage, at 43.0% and 26.5%, respectively. With Qwen3.5, SkillDyG reaches 43.8% P@3 and 29.7% P@5, again exceeding the corresponding strongest baseline at 40.2% and 25.2%. These results show that SkillDyG can progressively form a retrievable capability organization from an open skill collection without relying on a reference organization or SkillRet query–skill relevance annotations. Its group boundaries, roles, and context help identify relevant skills in large candidate collections.

G Additional Ablation Studies (RQ2)

G.1 Existing-skill Validation Policies

For the detailed sampling analysis, we use one maintenance update as a common snapshot after shared initialization. The table reports the settings visualized in Figure 6 together with the 100% validation reference. At each candidate update, every newly added skill is validated, while existing members are sampled according to the weighted rule in Appendix E.3; the construction and validation prompts do not contain evaluation labels. For the organization scores in this table, we use each skill's source-task category annotation as the external reference partition; for skills associated with multiple source categories, the primary source-task category is used. These annotations are used only by the offline scorer. We do not directly use task gold-skill sets to define a single reference partition because a skill can occur in the gold sets of multiple tasks, making the resulting relation overlapping rather than a mutually exclusive partition; applying V-measure and ARI directly to such labels would violate their single-label partition assumption. Pairwise precision, recall, and F_1 based on gold-skill co-occurrence are a more suitable complementary evaluation for that setting.

Table 7: Retrieval metrics for component ablations of SkillDyG on SkillsBench. We construct a 252-skill library over 14 arrival rounds and evaluate retrieval on 95 tasks. Prec. and Rec. are macro averages. F_1 is their harmonic mean.

Configuration	Prec.	Rec.	F_1
SkillDyG	**91.22**	**90.67**	**90.94**
w/o Source-Context Denoising	65.08	63.78	64.42
w/o Group Boundary Self-Evolution	83.58	82.33	82.95
w/o Self-Evolution Validation	82.18	80.83	81.50
Local-only Retrieval	85.39	81.87	83.59
Global-only Retrieval	74.61	73.72	74.16

Table 8: Detailed existing-member sampling analysis on SkillsBench at a common maintenance update. Consumed tokens denote the normalized input at the common snapshot. Classification F_1 is macro-averaged after Hungarian matching between predicted groups and source-task categories.

Sampling strategy	Old members checked	Consumed tokens ↓	B^3 F_1 ↑	V-measure F_1 ↑	ARI ↑	Classification F_1 ↑
0%	0	5.81K	0.288	0.657	0.113	0.481
25%	16	6.35K	0.270	0.659	0.084	0.466
50%	32	7.29K	0.354	0.687	0.197	0.542
100%	64	7.51K	0.362	0.696	0.201	0.569
Our Method	7	6.18K	0.360	0.690	0.211	0.533

Table 9: Fixed Top-K retrieval on the shared SkillsBench pilot. K fixes the returned selection size; Our Method selects its size adaptively. Precision, recall, F_1, and full coverage are computed against fixed task gold-skill annotations used only for retrieval scoring. Full coverage is the percentage of tasks whose complete gold-skill set is selected.

Budget	Retrieval view	P ↑	R ↑	F1 ↑	Full coverage (%) ↑	Avg. selected skills
$K = 1$	Dual	96.7	45.7	57.5	20.0	1.00
	Local	100.0	46.6	**58.8** (+2.3%)	20.0	1.00
	Global	90.0	43.5	54.1	20.0	1.00
$K = 2$	Dual	71.7	60.8	60.9	30.0	2.00
	Local	75.0	62.8	**63.3** (+3.9%)	30.0	2.00
	Global	71.7	59.6	59.9	26.7	2.00
$K = 3$	Dual	66.7	78.0	**67.1** (+0.6%)	56.7	3.00
	Local	64.4	75.3	64.7	53.3	3.00
	Global	66.7	77.5	66.7	56.7	3.00
$K = 4$	Dual	57.5	85.3	**64.3** (+3.4%)	70.0	4.00
	Local	55.8	82.7	62.2	66.7	4.00
	Global	54.2	81.5	60.7	63.3	4.00
$K = 5$	Dual	48.7	87.3	**58.4** (+5.2%)	70.0	5.00
	Local	46.0	83.8	55.5	66.7	5.00
	Global	45.3	83.4	55.0	66.7	5.00
$K = 6$	Dual	41.7	88.3	**53.1** (+2.3%)	73.3	6.00
	Local	40.0	85.7	51.1	70.0	6.00
	Global	40.6	86.9	51.9	70.0	6.00
Our Method	Dual	76.6	82.6	**77.2** (+2.1%)	66.7	3.17
	Local	73.9	83.6	75.6	70.0	3.30
	Global	67.2	77.7	69.5	60.0	3.27

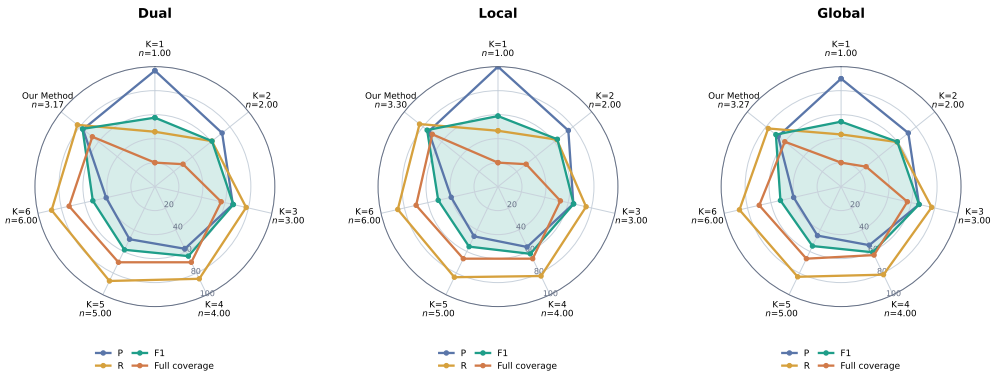


Figure 13: Detailed fixed Top-K retrieval profiles on SkillsBench. Each panel corresponds to a retrieval view. Spokes give the budget and average number of selected skills; the green shaded region marks F1, while the other curves report precision, recall, and Full coverage.

Table 10: Interference resistance on 32 SkillsBench tasks whose gold skills are complete from round 8, reevaluated as the library grows from 128 to 219 skills. Scores are percentages. Best result in each column and checkpoint is in bold.

Round	Skills	Method	P@3	P@5	Retrieval F_1
8	128	SkillDyG	**54.2** (+10.6%)	**34.4** (+7.8%)	**93.8** (+0.1%)
8	128	Vector Skills	35.4	24.4	78.7
8	128	GoS	49.0	31.3	93.1
8	128	GoSkills	47.9	31.9	93.7
10	160	SkillDyG	**54.2** (+10.6%)	**34.4** (+7.8%)	**93.7** (+1.6%)
10	160	Vector Skills	35.4	22.5	77.8
10	160	GoS	49.0	31.3	92.2
10	160	GoSkills	47.9	31.9	91.6
12	192	SkillDyG	**54.2** (+10.6%)	**34.4** (+7.8%)	**93.7** (+3.2%)
12	192	Vector Skills	30.2	20.6	74.0
12	192	GoS	49.0	31.3	89.1
12	192	GoSkills	47.9	31.9	90.8
14	219	SkillDyG	**54.2** (+15.6%)	**34.4** (+7.8%)	**93.2** (+4.5%)
14	219	Vector Skills	32.3	20.6	73.4
14	219	GoS	44.8	31.3	79.8
14	219	GoSkills	46.9	31.9	89.2

H Additional Model Studies(RQ3)

H.1 Retrieval Trajectories under Skill-Library Expansion

We fix two SkillsBench task cohorts before scoring and do not reselect them after later rounds. The interference study uses the 32 tasks whose gold skills are all present in the round-8 library S_8. The admission study uses the 40 tasks with a nonempty but incomplete gold set at round 10. Skills arrive in batches of at most 16. We evaluate the interference cohort at rounds 8, 10, 12, and 14 ($|S_t| \in \{128, 160, 192, 219\}$) and the admission cohort at rounds 10, 12, and 14. Gold labels are used only to form the cohorts and to score outputs; they are not shown in retrieval prompts. This cohort design differs from Table 2, which evaluates all 95 SkillsBench tasks once against the complete 219-skill library. RQ3 instead reevaluates fixed 32- and 40-task cohorts across intermediate library states, so the two sets of results are not directly comparable.

Let G_q denote the final gold set of task q and $A_{q,t} = G_q \cap S_t$ the gold skills already in the current library. Alternate identifiers of the same gold instance are counted once. P@k is the number of distinct elements of $A_{q,t}$ among the first k ranks, divided by k; unfilled ranks count as misses. For the adaptively selected set, we macro-average precision and recall over tasks against $A_{q,t}$, then compute Retrieval F_1 as their harmonic mean, matching the aggregation used for the component ablations in Appendix G. Gold availability is the cohort mean of $|A_{q,t}|/|G_q|$, and the P@k ceiling is $\mathrm{mean}_q \min(|A_{q,t}|, k)/k$. A later-round increase in P@k can therefore reflect a larger $A_{q,t}$ rather than better ranking, so methods are compared within a round. A perfect current-round F_1 does not imply that the final set G_q has been recovered. The two trajectories are reported in Tables 10 and 11.

Table 11: Skill admission: retrieval on 40 SkillsBench tasks with incomplete gold at round 10. Gold available is the mean fraction of each task's gold skills present in the current library; 0, 13, and 40 tasks have complete gold at rounds 10, 12, and 14. Compare methods within a round. Scores are percentages. Best available result in each column and checkpoint is in bold.

Round	Skills	Gold available	P@3 ceiling	P@5 ceiling	Method	P@3	P@5	Retrieval F_1
10	160	57.6	62.5	41.5	SkillDyG	**57.5** (+15.0%)	**39.5** (+21.5%)	**83.7** (+2.3%)
10	160	57.6	62.5	41.5	Vector Skills	32.5	21.5	61.8
10	160	57.6	62.5	41.5	GoS	49.2	32.0	80.6
10	160	57.6	62.5	41.5	GoSkills	50.0	32.5	81.8
12	192	73.6	72.5	52.0	SkillDyG	**65.8** (+25.3%)	**48.0** (+23.1%)	**83.1** (+3.5%)
12	192	73.6	72.5	52.0	Vector Skills	43.3	27.0	54.7
12	192	73.6	72.5	52.0	GoS	52.5	38.0	75.1
12	192	73.6	72.5	52.0	GoSkills	50.8	39.0	80.3
14	219	100.0	90.0	66.5	SkillDyG	**72.8** (+7.9%)	**60.1** (+20.2%)	**79.8** (+1.0%)
14	219	100.0	90.0	66.5	Vector Skills	54.2	36.0	51.2
14	219	100.0	90.0	66.5	GoS	67.5	48.5	70.9
14	219	100.0	90.0	66.5	GoSkills	67.5	50.0	79.0

I Prompt Templates and Case Study

This appendix gives the prompt templates for the three core decisions in continual skill-library construction and for dual-head retrieval. Each box contains the system instruction followed by the runtime payload and output contract; angle-bracketed fields are filled from the current library state. We then follow one group as it evolves from a single search method into a reusable workflow organization.

I.1 Prompt Templates

P1: New-Skill Admission. **Purpose.** P1 sees the local view of one normalized candidate and the frozen catalog of groups. It prepares a membership application with a grounded rationale but cannot modify the group state or produce an executor-facing selection. **Decision.** It either applies to an existing group or proposes a new group, and records any supported role or workflow relation.

P1: New-Skill Admission

```
SYSTEM
You process one normalized skill candidate from the current batch.
    The batch is completed
before retrieval is served, so do not alter the committed group state
     or produce a retrieval selection.
The skill's local view describes only itself. The group catalog is
    the entire
currently committed library, frozen for this batch. Do not infer
    anything from
future skills. Source-task provenance is intentionally withheld from
    routing.

Choose an existing group only when its membership boundary and
    current members
support the skill's concrete capability, artifact, tool, check, or
    constraint.
If no current group fits, propose a concise new group. A skill may
    apply to
more than one group only when shared or conditional membership has a
    concrete
reason. Do not output scores or confidence values.

Every application needs a short grounded reason. A proposed colleague
     role or
```

```
relation is useful when evidence supports it; otherwise use role "
    unknown" and
an empty relation list. Never invent a workflow merely to fill a
    field.
Return exactly one JSON object.

USER
{"new_skill_local_view":{"event_id":"<event id>","skill_name":"<skill
     name>",
 "known_library_skill_id":"<optional canonical id>","summary":"<local
      summary>",
 "capabilities":["<operation>"],"artifacts":["<input/output artifact
     >"],
 "tools":["<tool cue>"],"check_hints":["<check>"],"constraint_hints
     ":["<constraint>"]},
 "active_group_catalog":[{"group_id":"<active group id>","name":"<
     group name>",
 "boundary":{"include":"<positive boundary>","exclude":"<negative
     boundary>",
 "stable_core":"<reusable core>"},"formal_members":["<member cards
     >"]}],
 "required_output":{"applications":[{"group_id":"<active id or
     NEW_GROUP>",
 "membership_scope":"primary | shared | conditional","reason":"<
     grounded reason>",
 "proposed_colleague_position":{"role":"<brief group-role description
     >",
 "relations":[["new_skill","feeds_into | verifies | alternative_to |
     incompatible_with",
 "existing_library_skill_id"]]},"new_group_name":"<only for NEW_GROUP
     >"}]}}
```

P2: Group Boundary Update. **Purpose.** P2 turns the applications from a completed batch into a candidate group definition. **Decision.** It gives every applicant an explicit action, updates the reusable boundary when necessary, and preserves existing structure unless the supplied evidence supports a change.

P2: Group Boundary Update

```
SYSTEM
You update one dynamic skill group after the current batch closes.
The old group definition is frozen history. The applications are soft
     reasons
from newly arrived skill candidates, not hard facts. Propose a
    candidate new boundary,
member treatment, roles, execution order, and verification relations.
     Keep the
boundary reusable: do not promote source-task-specific details to the
     stable
core merely because they appear in one incoming skill.

Use an explicit action for every applicant. Preserve existing member
    structure
unless the supplied evidence justifies a change. A missing colleague
    placement
is acceptable; do not fabricate edges. Do not output confidence
    scores.
Return exactly one JSON object.

USER
```

```
{"old_group_definition":{"group_id":"<group id>","name":"<group name
    >",
 "boundary":{"include":"<old include>","exclude":"<old exclude>",
 "stable_core":"<old stable core>"},"formal_members":["<members>"],
 "workflow_edges":["<relations>"]},
 "incoming_applications":[{"event_id":"<event id>","library_skill_id
    ":"<canonical id>",
 "skill_name":"<skill name>","summary":"<summary>","capabilities
    ":["..."],
 "artifacts":["..."],"tools":["..."],"check_hints":["..."],"
    constraint_hints":["..."],
 "membership_scope":"primary | shared | conditional","
    application_reason":"<P1 reason>",
 "proposed_colleague_position":{"role":"<brief group-role description
    >",
 "relations":[["...","...","..."] ]}}],
 "repair_request":{"failure_source":"application | boundary | role |
    relation | none",
 "action":"rewrite_boundary | change_membership | split |
    create_group",
 "reason":"<provided after validation failure>"},
 "required_output":{"candidate_group":{"name":"<short name>",
 "boundary":{"include":"<positive boundary>","exclude":"<negative
    boundary>",
 "stable_core":"<reusable core>"}},
 "candidate_member_actions":[{"event_id":"<incoming event id>",
 "action":"admit | shared | conditional | create_group",
 "membership_scope":"primary | shared | conditional",
 "role":"<brief group-role description>","reason":"<short explanation
    >"}],
 "candidate_workflow_edges":[["library_skill_id",
 "feeds_into | verifies | alternative_to | incompatible_with","
    library_skill_id"]],
 "update_reason":"<short explanation>"}}
```

P3: Existing-Member Validation and Repair. **Purpose.** P3 evaluates every arriving applicant and a sampled set of established members against the candidate boundary, roles, and workflow relations. **Decision.** The established-member sample prioritizes newer members and compensates members that have gone longer without audit. A failure is classified as an application, boundary, role, or relation error, which determines which component is repaired before the candidate organization is validated again.

P3: Existing-Member Validation and Repair

```
SYSTEM
You validate one candidate update to a dynamic skill group.
Check every incoming applicant and the supplied sample of old formal
members against the candidate boundary, member roles, and workflow
    relations.
The question is whether this candidate group can consistently contain
     the
members it claims to contain. A pass must account for every listed
    applicant and
formal sample. If the candidate fails, identify whether the error
    lies in an
applicant's application, the group boundary, a role, or a relation,
    and return a
repair request for the corresponding component.
Do not produce confidence scores and do not invent missing evidence.
Return exactly one JSON object.
```

```
USER
{"candidate_group":"<P2 group>","candidate_member_actions":["<P2
    actions>"],
 "candidate_workflow_edges":["<P2 edges>"],
 "all_interns":[{"event_id":"<event id>","library_skill_id":"<
    canonical id>",
 "skill_name":"<skill name>","summary":"<summary>","capabilities
    ":["..."],
 "artifacts":["..."],"tools":["..."],"check_hints":["..."],"
    constraint_hints":["..."],
 "application_reason":"<P1 reason>",
 "candidate_action":"admit | shared | conditional | create_group",
 "candidate_role":"<brief group-role description>"}],
 "formal_member_sample":["<sampled formal-member cards>"],
 "required_output":{"result":"pass | repair_required",
 "intern_results":[{"event_id":"...","valid":true,"reason":"..."}],
 "formal_sample_results":[{"library_skill_id":"...","valid":true,"
    reason":"..."}],
 "violations":["..."],"repair_request":{"failure_source":
 "application | boundary | role | relation","action":
 "revise_application | rewrite_boundary | change_membership | split |
     create_group","reason":"..."}}}
```

Validation commits the updated organization only after it passes. An application error returns the affected candidate to P1, whereas a boundary, role, or relation error returns the affected group to P2. The repaired candidate organization is then validated again before retrieval is served.

P4: Dual-Head Retrieval and Fusion. **Purpose.** P4 first recalls candidates from local capability cards and global group cards, then selects a compact, coverage-complete set from their union. **Decision.** The first stage deliberately keeps local and global evidence separate; the second stage assigns roles and records requirement coverage for the final selection.

P4a: Dual-Head Recall

```
SYSTEM
You are the recall stage of a two-stage skill retriever.
Use only the visible query and supplied cards. Never infer hidden
    tests or gold labels.
The current batch has completed maintenance before retrieval begins.
    Local cards come from
the completed skill library, while global cards come from the
    committed group state.
Maintain two genuinely different heads:
- local_head matches concrete operations, artifacts, tools,
    constraints, and checks.
- global_head matches reusable group shared cores and group
    boundaries.
Optimize recall, not precision. Include plausible alternatives when
    wording differs.
Do not select a skill merely because its source domain is adjacent.
    Return one JSON object.

USER
{"global_group_cards":[{"group_id":"<id>","name":"<group name>",
 "shared_task_core":"<group core>","operation_cues":["..."],"
    artifact_cues":["..."],
 "positive_boundary":["..."],"negative_boundary":["..."],"
    member_occurrence_ids":["..."]}],
 "local_skill_cards":[{"occurrence_id":"<id>","skill_id":"<id>",
```

```
 "capability":"<concrete capability>","operation_cues":["..."],"
    artifact_cues":["..."],
 "tool_cues":["..."],"required_checks":["..."]}],"query":"<task query
    >",
 "limits":{"local_head":"<k>","global_groups":"<k>","global_skills
    ":"<k>","union":"<k>"},
 "output_contract":{"local_head":["ordered occurrence ids"],
 "global_head":{"group_ids":["ordered group ids"],"occurrence_ids":["
    ordered occurrence ids"]},
 "recall_union_occurrence_ids":["ordered occurrence ids"]}}
```

P4b: Precision Selection

```
SYSTEM
You are the precision stage of a two-stage skill retriever.
Use only the visible query and supplied Stage-1 candidates. Build the
     smallest
coverage-complete complementary expert set, not merely the shortest
    list.

First decompose every explicit deliverable, operation, constraint,
    and validation
obligation in the query into atomic requirements. Mark a requirement
    critical when
omitting it could make the visible task incorrect. Judge every
    candidate against
those requirement IDs. A skill may be dropped as redundant only when
    another selected
skill covers the same requirements at equal or better specificity.
    Same-domain
similarity alone is never evidence. Do not collapse method selection,
     API use,
artifact transformation, and validation.

Judge birth-scene information query-conditionally. Prefer concrete
    operation+artifact
fit over source-domain similarity. Assign an execution role to each
    retained skill.
Every critical requirement must be covered when any candidate covers
    it. Return one
JSON object. Never use hidden tests or gold labels.

USER
{"query":"<task query>","max_selected":"<maximum selected skills>",
 "selection_objective":"coverage of critical requirements is hard;
    compactness is secondary",
 "candidate_policy":"return one decision for every candidate;
    uncovered critical requirements override a drop",
 "candidates":["<Stage-1 candidate cards>"],
 "output_contract":{"atomic_requirements":[{"requirement_id":"R1",
 "description":"<explicit obligation>","requirement_type":"operation
    | artifact | constraint | validation",
 "critical":true}],"candidate_decisions":[{"occurrence_id":"<id>",
 "decision":"keep | drop","role":"<brief execution-role description
    >",
 "covers_requirement_ids":["R1"],"query_fit":"0..1","transferability
    ":"0..1",
 "birth_bias_risk":"0..1","redundancy_cluster":"<label or null>","
    reason":"<evidence>"}],
 "selected_occurrence_ids":["ordered ids, at most max_selected"],
```

```
"coverage_audit":{"R1":["selected ids"]},"selection_summary":"<short
    summary>"}}
```

I.2 Illustrative Boundary-Repair Case

Case. Consider a task that asks to prepare a stellar light curve for transit analysis, search for recurring transit-shaped dimming events, inspect aliases and artifacts, and produce a candidate report with the checks required before interpretation.

Initial group and evolution. At round 1, `box-transit-period-search` contains only Box Least Squares (BLS). Its stable core is detecting and validating periodic box-shaped dimming events in light curves; it excludes sinusoidal periodograms, full physical transit fitting, spectroscopy, and unrelated time-series anomaly detection. At round 3, an end-to-end exoplanet light-curve workflow expands this BLS-centered group to `exoplanet-light-curve-workflows`, adding method selection, candidate inspection, artifact checks, validation, and characterization planning. At round 6, light-curve preprocessing is admitted solely as a preparation stage for transit-oriented analysis; sinusoidal-only search remains excluded. At round 13, Transit Least Squares (TLS) is admitted as a complementary transit-search method.

Round	Structural decision
1	A BLS-only group is initialized for periodic box-shaped dimming events.
3	The boundary broadens to an end-to-end exoplanet light-curve workflow.
6	Preprocessing is admitted as a preparation stage; sinusoidal-only search stays outside.
13	TLS is admitted as a complementary transit-search method.

Admission, update, and validation. For the round-6 arrival, the local view of *Light-Curve Preprocessing* identifies quality-flag handling, trend removal, flattening, and uncertainty alignment. It maps a raw light curve to a cleaned one while preserving transit-shaped signals. P1 proposes membership in `exoplanet-light-curve-workflows` because the cleaned curve is consumed by downstream search and candidate-validation stages. P2 admits the skill as a preparation-stage member, adds these operations to the positive boundary, and retains the exclusions for general-purpose cleaning and sinusoidal-only search. P3 checks the new member together with BLS against the revised boundary. The candidate passes because preprocessing supplies a required workflow input without turning BLS into a general time-series method.

Retrieval after the update. After round 13, P4 decomposes the task into preparing the light curve without removing transit evidence, finding recurring transit-shaped dimming, checking alias and artifact explanations, and producing a candidate report. Its selected set contains *Light-Curve Preprocessing*, *Box Least Squares*, *Transit Least Squares*, and *Exoplanet Workflows*. This set covers preparation, complementary search, validation, and reporting. A sinusoidal-only period-search skill is excluded: both the query and the group boundary rule out this otherwise domain-adjacent alternative.

I.3 Error Case

Analytic measurement versus media transformation. Figure 14 shows an over-broad admission caused by shared input modality. In the saved round-3 GoSkills organization, *Energy Calculator* is placed in `media-audio-video-processing` alongside audio extraction, FFmpeg processing, and format-conversion skills. These skills may all consume audio, but their operations and output artifacts differ. The established group transforms media streams or recovers media content. *Energy Calculator* outputs an RMS-energy statistic. Our boundary check therefore retains it as a separate analytic capability rather than a member of the media-transformation group. This case shows that a shared input domain is not sufficient for continual admission. The group boundary must also constrain the operation and output artifact of its members.

Security scoring versus browser-quality testing. Figure 15 shows a second error caused by a broad functional label. The baseline `engineering-investigation-and-validation`

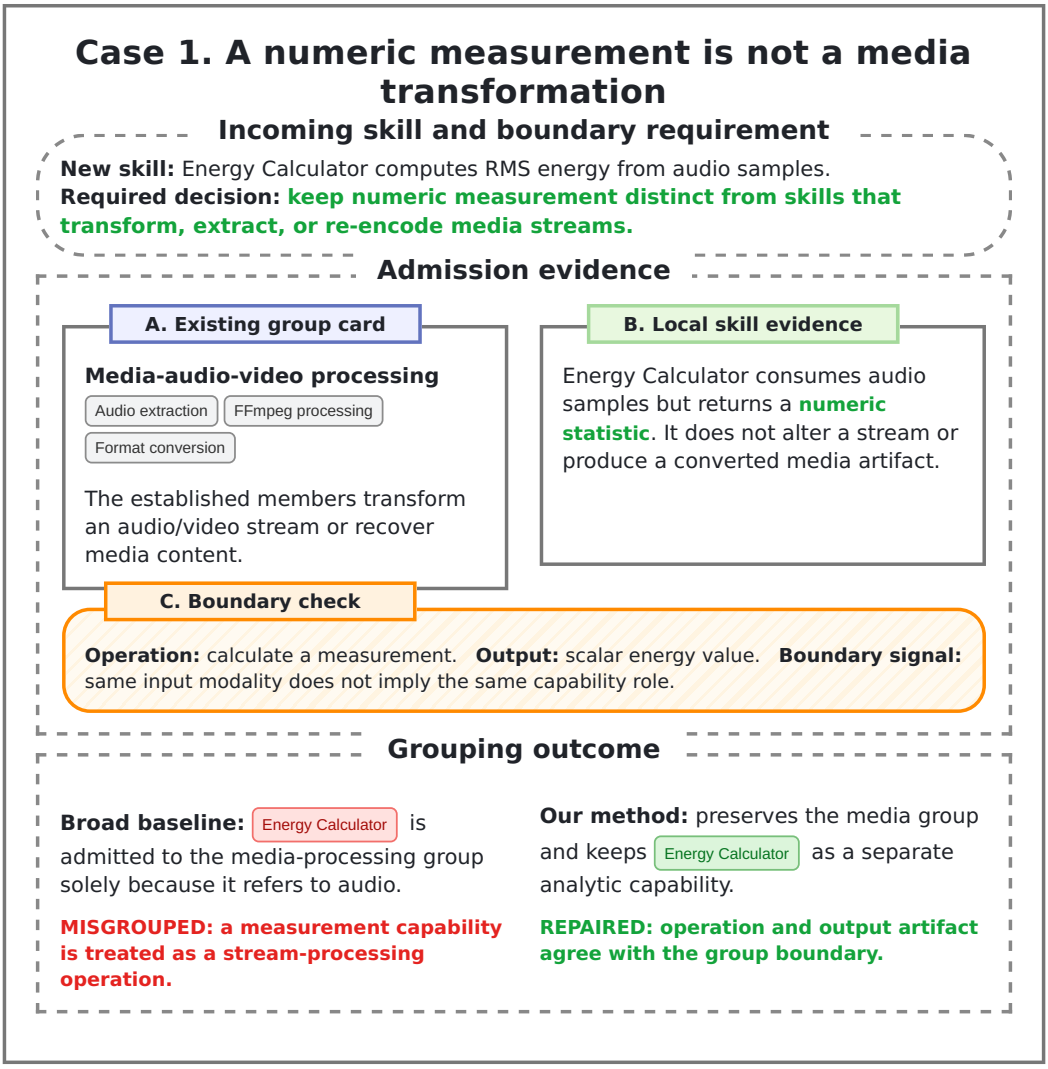


Figure 14: Error case: Analytic measurement versus media transformation.

group contains both *CVSS Score Extraction* and *Browser Performance Visual Stability Testing*. Both can be described as software validation, but they support different decisions and operate over different artifacts. CVSS extraction reads vulnerability records for security auditing and reporting. Visual-stability testing analyzes browser rendering traces for page-quality assessment. Our method groups CVSS extraction with vulnerability scanning and security-reporting skills, while retaining browser-quality testing in a separate workflow. This example shows that broad group names can hide incompatible task roles. Preserving artifact and reporting-objective boundaries avoids this conflation.

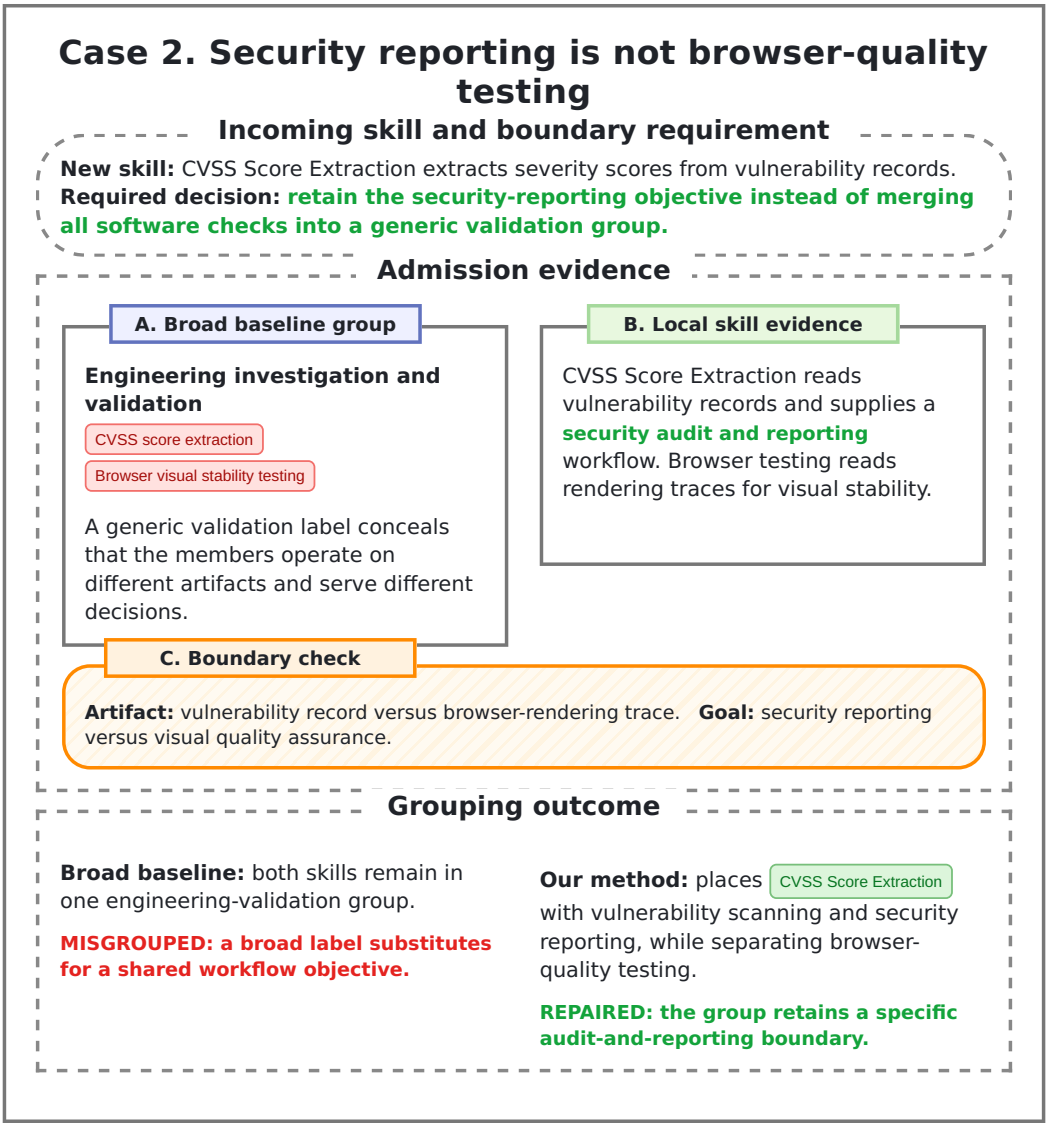


Figure 15: Error case: Security scoring versus browser-quality testing.

J The Use of Large Language Models (LLMs)

J.1 Writing and Formatting Assistance

In the preparation of this manuscript, we used a large language model as a writing assistant. Its main role was to help improve our English writing, such as correcting grammar and refining sentences for clarity and style. Additionally, it was used to help set up the initial format for several tables. The authors made all final decisions on the content, carefully checking and editing all suggestions from the model to ensure the scientific accuracy and integrity of this work.