SKILLSWIKI: TAMING EVER-GROWING LLM AGENT SKILLS WITH A SELF-EVOLVING HIERARCHICAL WIKI

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model agents increasingly rely on a large and diverse repertoire of skills to execute long-horizon tasks, yet existing frameworks manage these skills as a flat, isolated, and static list retrieved by embedding similarity. This design scales poorly as the skill library grows, ignores the relational structure among skills, and discards the experience accumulated across tasks. We propose SkillsWiki, a unified framework that organizes agent skills into a three-layer wiki. A skill taxonomy constructed by large language models materializes skill categories as a hierarchical directory tree with summarization nodes, enabling coarse-to-fine retrieval. A skill relation graph, autonomously discovered by the agent through wiki exploration, links related skills and couples a shared cloud-side wiki with a local workspace wiki. A skill package library distills proven skill compositions from historical execution traces and surfaces them only when an associated skill is activated, enabling context-triggered reuse. The entire structure is maintained as a lightweight, file-based wiki that remains readable and editable by human developers. Experiments on three agent benchmarks with multiple backbone models show that SkillsWiki consistently improves task success rate and reduces execution steps over strong baselines.

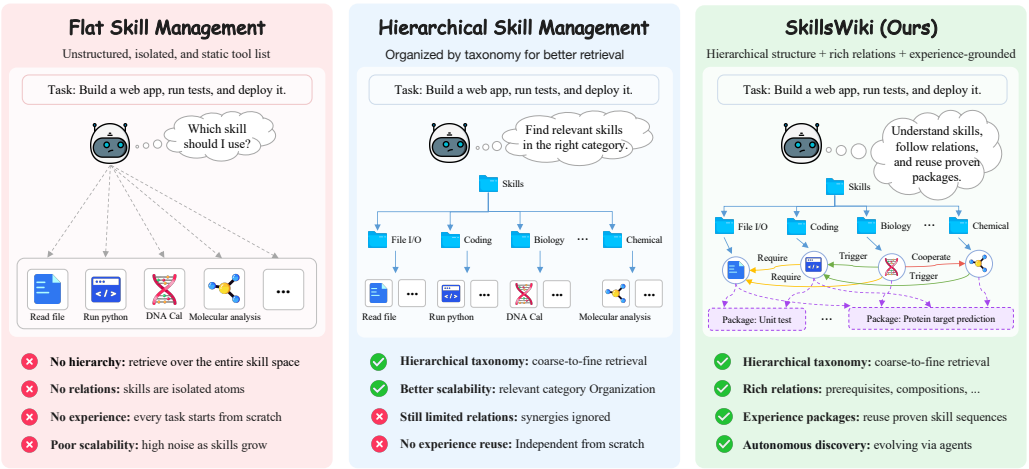


Figure 1: **Evolution of skill management for LLM agents.** Flat lists scale poorly and ignore relations and experience, while hierarchy alone leaves skills isolated. SkillsWiki unifies taxonomy, relations, and experience-grounded packages into one autonomously evolving wiki.

1 INTRODUCTION

Large language model (LLM) agents have rapidly evolved from single-turn question answering systems into autonomous entities capable of executing long-horizon, multi-step tasks spanning software

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.

engineering, scientific discovery, web navigation, and robotic planning (Yao et al., 2022b; Shinn et al., 2023; Hong et al., 2024). Central to this evolution is the agent's skill repertoire, the collection of callable tools, APIs, code snippets, and procedural routines that it can invoke to interact with external environments (Wu et al., 2026a). As the scope of agent applications broadens, the number and diversity of required skills grow combinatorially (Peng et al., 2019). For instance, a single coding agent may need to orchestrate file I/O, version control, compilation, testing, and deployment skills, each with multiple variants and contextual preconditions.

Despite this growing complexity, the prevailing paradigm for skill management remains strikingly flat, isolated, and static (Li, 2026). Most existing tool-use frameworks (Qin et al., 2024; Patil et al., 2024; Shen et al., 2023) represent the available skill set as an unstructured list of API descriptions, retrieved via embedding similarity at inference time. This design suffers from three fundamental limitations. First, the absence of a hierarchical taxonomy means that retrieval must operate over the entire skill space, leading to high noise and poor scalability as the skill library grows. Second, skills are treated as independent atoms; the rich relational structure among them, including prerequisite dependencies, compositional synergies, and functional substitutions, is entirely ignored, preventing the agent from reasoning about which skills to combine and in what order. Third, there is no mechanism for experiential accumulation: every new task triggers exploration from scratch, even when structurally similar skill sequences have been successfully executed before.

Recent works have explored structured skill representations and experience reuse from complementary perspectives. Some methods (Liang et al., 2026; Su et al., 2026; Zeng et al., 2026) learn skill representations over a fixed, pre-defined skill inventory, which supports structured composition but does not allow autonomous discovery of new skills or their inter-relations Memory-augmented agents (Wang et al., 2023; Chhikara et al., 2025) store past trajectories but lack a structured index that links historical experience to the skill taxonomy, making targeted reuse difficult. In short, no existing framework jointly addresses (i) structured skill organization, (ii) autonomous skill-relation discovery, and (iii) experience-grounded skill packaging in a unified, scalable architecture.

We draw inspiration from a remarkably successful human solution to the analogous problem of organizing vast, heterogeneous knowledge: the wiki. A wiki provides a natural taxonomy through nested categories, encodes relational knowledge via cross-reference links, and supports progressive exploration, so readers discover new topics by following links from what they already know. Crucially, a wiki is simultaneously human-readable and machine-navigable, making it an ideal substrate for both developer oversight and agent autonomy.

Based on this idea, we propose SkillsWiki, a hierarchical wiki-structured skill repository for LLM agents. SkillsWiki contains three complementary layers: 1) *Skill Taxonomy*, a wiki of skill categories constructed by LLMs and materialized as a hierarchical directory tree with summarization nodes, enabling coarse-to-fine retrieval; 2) *Skill Relation Graph*, an evolving network of inter-skill links autonomously discovered by the agent through exploration, distinguishing cloud-side general relations among shared skills from local-side plugin relations that bind specialized, environment-specific skills to their general counterparts; and 3) *Skill Package Library*, a set of experience-grounded skill compositions distilled from historical execution traces, stored under the corresponding skill folders and surfaced only when the associated skill is activated, thereby enabling progressive, context-triggered reuse. The entire structure is maintained as a lightweight, file-based Wiki that supports metadata indexing for retrieval while remaining transparent and editable by human developers.

Our main contributions are as follows:

- A principled skill organization framework. We formalize the skill space as a three-layer wiki architecture (Taxonomy $\rightarrow$ Relation Graph $\rightarrow$ Package Library) that organizes skills for structured discovery, relational navigation, and experiential reuse.
- Autonomous skill-relation discovery. We design an agent-driven exploration protocol that incrementally constructs the Skill Relation Graph, including cloud and local cross-links, without manual annotation.
- Experience-grounded skill packaging. We introduce Skill Packages that encapsulate proven skill compositions from historical tasks, enabling context-triggered reuse that reduces redundant exploration and lowers execution steps.
- Extensive empirical validation. We evaluate SkillsWiki on three agent benchmarks, demonstrating gains in average reward and execution steps over strong baselines.

2 Related Work

Structured Skill Organization and Retrieval. Prompt-based agents like ReAct (Yao et al., 2022b) treat skills as flat context lists, causing retrieval noise at scale. Structured alternatives, including HiSkill's hierarchical indexing (Hao et al., 2026), SkillScope's contextual filtering (Wu et al., 2026b), SkillRouter's learned routing (Zheng et al., 2026), ToolLLM's API-aware retrieval (Qin et al., 2024), and related group-structured (Zeng et al., 2026) or retrieval-augmented (Su et al., 2026) methods, improve precision but assume static, pre-defined taxonomies requiring manual curation. SkillsWiki dynamically builds a wiki-structured taxonomy, which supports hybrid cloud-local architecture, enabling autonomous scaling beyond static or prompt-only methods.

Skill Relation Modeling and Composition. Multi-step execution requires modeling inter-skill dependencies. SkillNet (Liang et al., 2026), learns latent compositions offline but assumes a fixed inventory; SkillGraph (Li et al., 2026) extracts edges from documentation yet similarly relies on offline mining; LATS (Zhou et al., 2023) plans over tool graphs but does not persist discovered relations. SkillsWiki incrementally builds the Skill Relation Graph via agent-driven relation discovery, separating general and plugin-specific relations for modularity.

Experience-Grounded Skill Reuse. Memory-augmented methods leverage past trajectories. Expel (Zhao et al., 2024) distills cross-task rules into a flat buffer but decouples them from skill structure. Reflexion (Shinn et al., 2023) stores episodic self-reflections without compositional grounding. Voyager (Wang et al., 2023) indexes a code library by description but lacks relational anchoring. Works such as Skill-DisCo (Guo et al., 2026) and Beyond Domains (He et al., 2026) move beyond description-based retrieval by using semantic clustering and transferable interaction patterns, yet neither anchors proven compositions within a structured skill directory. SkillsWiki addresses this via skill packages that anchor proven compositions and surface them lazily upon activation, enabling context-triggered reuse aligned with the taxonomy.

3 Preliminaries

We model an LLM agent's skill knowledge base as a *SkillsWiki*, a file-based, human-readable and machine-navigable wiki. Let $\mathcal{S} = \{s_1, \ldots, s_n\}$ be the set of skills, where each skill $s \in \mathcal{S}$ is a callable unit described by metadata $m(s) = (\mathrm{name}, \mathrm{signature}, \mathrm{desc}, \mathrm{io}, \mathrm{pre}, \mathrm{post})$, with pre and post denoting preconditions and postconditions. A SkillsWiki is a tuple

$$\mathcal{W} = (\mathcal{T}, \mathcal{G}, \mathcal{P}, \mathcal{I}),$$

where $\mathcal{T}$ is the skill taxonomy, $\mathcal{G}$ is the skill relation graph, $\mathcal{P}$ is the skill package library, and $\mathcal{I}$ is a metadata index for retrieval. The wiki pages are categories, skills, and packages, and the wiki links are taxonomy edges, relation edges, and package attachments.

Skill taxonomy. The taxonomy is a rooted tree

$$\mathcal{T} = (\mathcal{C}, \mathcal{E}_{\mathcal{T}}, \sigma, \lambda),$$

where $\mathcal{C}$ is the set of category nodes with root c_0, $\mathcal{E}_{\mathcal{T}} \subseteq \mathcal{C} \times \mathcal{C}$ are parent–child edges, $\sigma(c)$ is a summary of category c, and $\lambda(c) \subseteq \mathcal{S}$ are skills directly attached to c. Every skill s has a category path $(c_0, c_{i_1}, \ldots, c_{i_k})$ from the root through nested categories, inducing a partial order over skills where ancestors are coarser and descendants finer.

Skill relation graph. Skills are not isolated atoms. Inter-skill references form an evolving directed multigraph

$$\mathcal{G} = (\mathcal{S}, \mathcal{E}_{\mathcal{G}}, \phi),$$

where each edge $e = (s_i, s_j) \in \mathcal{E}_{\mathcal{G}}$ indicates that s_j is related to s_i and $\phi(e)$ is a natural-language justification that records the description of the relation and the reason it was formed. An activated skill is therefore associated with its prerequisites, alternatives, and composable neighbors through its incident edges. The graph is self-evolving, since it is refined by skill exploration and discovery:

$$\mathcal{G} \leftarrow \mathrm{Update}(\mathcal{G}, \mathcal{S}'),$$

where $\mathcal{S}'$ denotes newly explored or discovered skills together with the relations found for them. As exploration proceeds, previously isolated skills acquire relations and existing edges are extended or revised, so the graph grows with the skill collection itself.

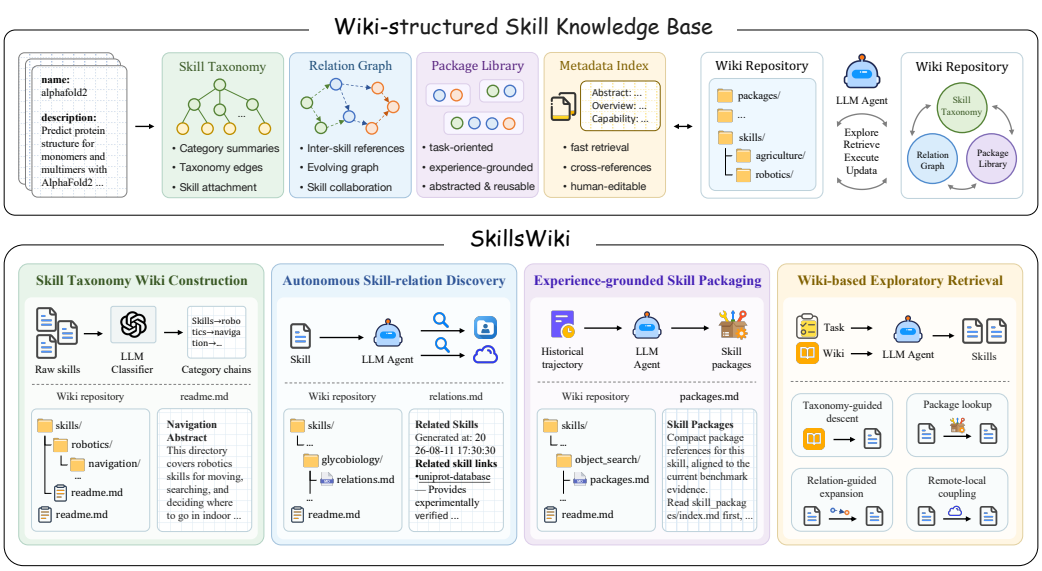


Figure 2: Overview of the Wiki-structured Skill Knowledge Base and the SkillsWiki framework, which integrates Skill Taxonomy Wiki Construction, Autonomous Skill-relation Discovery, Experience-grounded Skill Packaging, and Wiki-based Exploratory Retrieval to support LLM agents in exploring, retrieving, and updating skills.

Experience-grounded skill packages. A skill package is a task-oriented, reusable composition over skills. Formally,

$$p = (\tau_p, \mathcal{S}_p, \alpha_p, \rho_p, \gamma_p) \in \mathcal{P},$$

where τ_p is the triggering context, $\mathcal{S}_p = (s_{a_1}, \ldots, s_{a_m})$ is the composition of skills involved in the package, partitioned into required skills $\mathcal{S}_p^{\text{req}}$ and optional skills $\mathcal{S}_p^{\text{opt}}$, α_p is a summary of the capability pattern, ρ_p is a collaboration strategy describing how the skills coordinate, and γ_p is a set of execution reminders and failure guards. Each package is attached to its associated skills and stored under their wiki folders. A package is deliberately abstract. It records which skills collaborate, in what role, under what conditions, and with what cautions, but it does not store raw trajectories, environment-specific identifiers, step-by-step action logs, or per-skill metadata such as signatures and preconditions.

In summary, $\mathcal{W}$ organizes agent skills along three complementary axes, taxonomy for organization, relation graph for relations, and packages for composition, all materialized as wiki pages and links that support both human inspection and autonomous agent operation.

4 SkillsWiki

SkillsWiki instantiates the wiki-structured knowledge base $\mathcal{W} = (\mathcal{T}, \mathcal{G}, \mathcal{P}, \mathcal{I})$ from Section 3 as a file-based repository rooted at a workspace $\mathcal{R}$, where every category, skill, relation, and package is a file. In Figure 2, the pipeline builds taxonomy $\mathcal{T}$ and relation graph $\mathcal{G}$ from raw skills $\mathcal{S}_{\text{raw}}$, distills the package library $\mathcal{P}$ from traces, and performs wiki-based exploratory retrieval at test time.

4.1 Skill Taxonomy Wiki Construction

The taxonomy $\mathcal{T} = (\mathcal{C}, \mathcal{E}_{\mathcal{T}}, \sigma, \lambda)$ is built by an LLM classifier that assigns each skill a category

$$\mathcal{C}_i = c_0, c_{i_1}, \ldots, c_{i_k} = \text{Classify}_{\text{LLM}}(s),$$

which determines its location $\texttt{skills}/c_{i_1}/\cdots/c_{i_k}/\text{slug}(s)$. The directory tree thus is the taxonomy, and coarse-to-fine retrieval corresponds to directory (Tao et al., 2026). Each internal category carries an LLM-generated summary $\sigma(c)$, which supplies the semantic content for progressive disclosure and conditions the classifier when placing new skills.

Raw skills are serialized into a manifest $\{(s.\text{name},\ s.\text{desc},\ s.\text{source})\}_{s\in\mathcal{S}_{\text{raw}}}$, and the classifier processes it in batches. Each skill is assigned either an extension of an existing chain or new child categories under an existing node, preventing the tree from degenerating into a flat label set. The chains create the hierarchy, after which the original skill packages are indexed into the corresponding leaves. Classification uses a persistent cache, so re-runs only process changed skills.

After the hierarchy is fixed, every internal category $c \in \mathcal{C}$ is summarized by an LLM conditioned on the skills attached to its subtree,

$$\sigma(c) = \text{Summarize}_{\text{LLM}}\big(\lambda(c), \{\sigma(c')\}_{c'\in\text{children}(c)}\big),$$

so that the summary of a node aggregates its attached skills together with the summaries of its children. Summaries are therefore generated bottom-up, and each $\sigma(c)$ supplies the semantic content supporting progressive disclosure, while also conditioning the classifier when placing new skills.

4.2 Autonomous Skill-relation Discovery

The relation graph $\mathcal{G} = (\mathcal{S}, \mathcal{E}_{\mathcal{G}}, \phi)$ is discovered by the agent rather than annotated manually. For each source skill $s \in \mathcal{S}$, its related skills and the justifications for those relations are produced by an agent that explores the wiki through category summaries and skill metadata,

$$\mathcal{E}_{\mathcal{G}}(s),\ \phi(s) = \text{Explore}_{\text{Agent}}\big(m(s), \pi_{\mathcal{T}}(s), \mathcal{T}\big),$$

where $\mathcal{E}_{\mathcal{G}}(s) = \{(s, s') \mid s' \in \mathcal{S}\}$ is the set of edges induced by s, $\phi(s)$ assigns each such edge a natural-language justification that records the description of the relation and the reason it was formed, $m(s)$ is the skill metadata, $\pi_{\mathcal{T}}(s)$ is its category path, and $\mathcal{T}$ is the current taxonomy that the agent navigates. Because the agent reads summaries and metadata rather than comparing embeddings, discovery instantiates the same progressive-disclosure structure used at inference time. Candidates are validated against skills that actually exist, self-references are discarded, and duplicates are collapsed by (s, s'). Each surviving edge and its justification are written back as both a structured record and a human-readable cross-reference, so the same relation can be consumed by the agent and audited by a developer.

The graph is therefore self-evolving, since each newly discovered skill becomes a source for the same query and may yield further edges. Idempotence is enforced by a per-skill signature of the inputs that produced its relations, so unchanged skills are skipped on later runs.

4.3 Experience-grounded Skill Packaging

Given the package definition in Section 3, the construction problem is to distill $\mathcal{P}$ from execution traces without storing the episodes themselves. Let $\mathcal{D} = \{\xi_1, \ldots, \xi_N\}$ denote a set of traces, each recording the task summary, reward, step count, applied skills, action sequence, and terminal observation. The library is induced by an agent that abstracts over these traces while remaining grounded in the current skill organization,

$$\mathcal{P} = \text{Distill}_{\text{Agent}}\big(\mathcal{D}, \mathcal{S}, \mathcal{T}, \mathcal{G}\big),$$

where $\mathcal{S}$, $\mathcal{T}$, and $\mathcal{G}$ supply the skill set, taxonomy, and relation graph that constrain which compositions are admissible.

Successful traces yield positive compositions together with their summaries, collaboration strategies, and reminders, whereas unsuccessful traces contribute failure guards that warn against recurring mistakes. Both are abstracted before being written, since environment-specific identifiers and step-level logs are removed, which renders a package reusable and distinguishes $\mathcal{P}$ from a trajectory store. The resulting packages form a compact experience hierarchy that records which skills collaborate, in what role, under what conditions, and with what cautions, so that the agent can reuse proven compositions and avoid known pitfalls on related tasks.

4.4 Wiki-based Exploratory Retrieval

At inference time, the agent interacts with $\mathcal{W}$ through wiki-based exploratory retrieval, a sequential decision process over taxonomy, relation graph, and package library, rather than flat embedding retrieval. Given a task x, retrieval maintains a working skill set $\mathcal{S}_x \subseteq \mathcal{S}$, initialized as $\mathcal{S}_x \leftarrow \emptyset$ and grown until the agent judges the task covered. Packages and relations both serve as candidate evidence, and the agent decides which associated skills enter $\mathcal{S}_x$ and in what order they are executed.

Taxonomy-guided descent. The taxonomy first localizes the relevant capability area, so retrieval does not scan the entire skill space. Starting at the root, the agent inspects the summaries $\sigma(c)$ of the current node's children and selects a subset at each level until reaching leaf categories, admitting the applicable skills,

$$\mathcal{S}_x \leftarrow \mathcal{S}_x \cup \{s \in \lambda(C_L) \mid \text{Applicable}(s, x)\}.$$

Since the descent is a path selection over $\mathcal{T}$, the nodes inspected per level are bounded by the branching of the frontier rather than by $|\mathcal{S}|$.

Package lookup. The package library then supplies task-level reuse. For an activated skill s, the agent retrieves the packages whose composition contains s,

$$\mathcal{S}_x \leftarrow \mathcal{S}_x \cup \{s' \in \mathcal{S}_p \mid p \in \mathcal{P}(s),\ \tau_p \models x,\ \text{Needed}(s', x)\},$$

where $\mathcal{P}(s) = \{p \in \mathcal{P} \mid s \in \mathcal{S}_p\}$. An adopted package offers its composition as a candidate, from which the agent admits only the skills it judges necessary. As packages are attached to their constituent skills, reuse cost is bounded by $|\mathcal{P}(s)|$ rather than by $|\mathcal{P}|$.

Relation-guided expansion. The relation graph then extends the search beyond the taxonomy path. For skills in $\mathcal{S}_x$, the agent inspects their outgoing edges and admits only those neighbors it judges relevant,

$$\mathcal{S}_x \leftarrow \mathcal{S}_x \cup \{s' \mid (s, s') \in \mathcal{E}_\mathcal{G},\ \text{Relevant}(s', x)\},$$

where each edge carries a justification $\phi(e)$ recording the relation and its reason. Relation records thus act as expansion hints the agent filters rather than edges to be absorbed wholesale.

Cloud-local coupling. The above retrieval operates on the local wiki. When it cannot cover the task, the agent may additionally explore a remote general-purpose wiki and select the categories containing skills relevant to the current working set,

$$\mathcal{C}_x^{\text{cloud}} = \{c \in \mathcal{C}^{\text{cloud}} \mid \lambda(c) \cap \mathcal{S}_x \neq \emptyset\},$$

materializing the skills under these categories into the local wiki, where they cooperate with the existing local skills. Only $\mathcal{C}_x^{\text{cloud}}$ is materialized, so the local footprint stays proportional to it rather than to the full cloud wiki.

5 Experimental Setup

5.1 Datasets and Baselines

We evaluate SkillsWiki on three interactive agent benchmarks spanning embodied household manipulation, web navigation, and scientific reasoning: ALFWorld (Shridhar et al., 2020), WebShop (Yao et al., 2022a), and ScienceWorld (Wang et al., 2022). For ALFWorld and ScienceWorld we report results on both the Seen and Unseen splits, which separate task instances whose types were observed during training from those that were not. WebShop is evaluated on a single split.

We compare SkillsWiki against three representative baselines. ReAct (Yao et al., 2022b) interleaves reasoning and acting without any structured skill organization. Expel (Zhao et al., 2024) derives reusable experience from past trajectories but does not organize skills into a taxonomy or relation graph. SkillNet (Liang et al., 2026) maintains a learned skill representation but assumes a fixed inventory and supports neither autonomous relation discovery nor experience-grounded packaging.

5.2 Metrics

We report two metrics for each dataset. **R** denotes the average reward, and **S** denotes the average number of environment interaction steps, where higher R and lower S indicate better performance. For ALFWorld and ScienceWorld we report R and S separately on the Seen and Unseen splits, and for WebShop we report both on its single split.

5.3 Implementation Details

All methods are evaluated with the Claude Code harness [1], using MiniMax-M2.7 (Chen et al., 2026), GLM-5.1 (GLM-5-Team et al., 2026), and GPT-5.4-mini [2] as backbone models, so that the effect of SkillsWiki can be measured across different model families. Each episode is capped at 30 environment steps, 120 agent turns, and 8000 thinking tokens per turn. SkillsWiki is built in three stages. The skill taxonomy and the skill relation graph are constructed from the raw skill set, and the skill package library is distilled from execution traces on the training split, as described in Section 4. More detailed experimental settings can be found in Appendix A.

6 Results and Analysis

6.1 Main Results

Table 1: **Main results.** Reported task score $R \uparrow$ and mean environment steps $S \downarrow$. Bold denotes the best result within each backbone and column.

Method	ALFWorld				WebShop		ScienceWorld			
	Seen		*Unseen*		*Evaluation*		*Seen*		*Unseen*	
	$R \uparrow$	$S \downarrow$	$R \uparrow$	$S \downarrow$	$R \uparrow$	$S \downarrow$	$R \uparrow$	$S \downarrow$	$R \uparrow$	$S \downarrow$
MiniMax-M2.7										
ReAct	53.96	21.05	50.38	22.24	26.67	23.33	65.20	15.30	64.69	15.63
ExpeL	47.14	22.59	44.78	21.86	34.21	23.41	69.04	15.29	70.01	14.98
SkillNet	77.86	15.72	72.18	16.63	31.01	24.22	77.84	**11.37**	76.85	**11.55**
SkillsWiki	**86.43**	**13.56**	**82.09**	**14.34**	**36.27**	**19.33**	**78.65**	13.96	**77.14**	14.47
GLM-5.1										
ReAct	89.86	12.87	89.31	12.80	25.00	26.75	81.21	15.90	82.65	16.62
ExpeL	93.57	11.84	90.30	13.04	38.57	24.50	75.77	15.45	73.65	16.22
SkillNet	91.43	10.08	93.28	11.07	45.85	**20.03**	86.46	**13.81**	83.68	**13.90**
SkillsWiki	**99.29**	**9.63**	**95.52**	**9.88**	**52.25**	21.47	**88.14**	15.11	**87.73**	15.43
GPT-5.4-mini										
ReAct	40.71	24.36	44.03	24.25	12.26	26.14	59.62	14.54	56.92	15.17
ExpeL	71.43	18.56	78.36	18.00	8.50	25.93	56.95	13.44	60.20	13.72
SkillNet	80.00	**15.04**	76.12	**15.71**	14.90	**24.17**	78.55	**12.36**	70.21	**11.56**
SkillsWiki	**88.57**	15.51	**89.55**	15.98	**17.13**	26.14	**85.09**	13.87	**85.37**	15.06

Table 1 reports the main comparison across three benchmarks and three backbone models. SkillsWiki attains the highest success rate on every dataset and split under all three backbones, and the lowest average step count in most settings. The gains are consistent rather than model-specific, and they extend to the Unseen splits, indicating that the benefit generalizes to task types not observed during package construction rather than reflecting memorized instances. The step counts show the same trend, and in the few settings where SkillsWiki is not the most step-efficient, the marginal cost is more than offset by a substantial gain in success rate. Overall, structured skill organization, relation-guided expansion, and experience-grounded reuse jointly improve both success and efficiency across model families and across seen and unseen distributions.

6.2 Ablation Study

Table 2 shows that each SkillsWiki layer contributes to performance, since removing any one layer reduces task success across all three benchmarks. The largest drop occurs without packages, indi-

[1] https://code.claude.com/docs

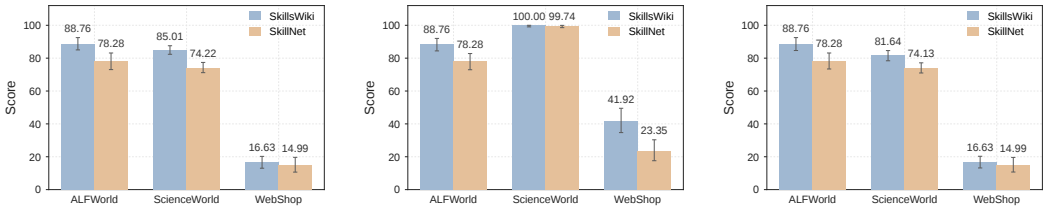
[2] https://developers.openai.com/api/docs/models/gpt-5.4-mini

Table 2: **Ablation study.** Impact of SkillsWiki components with Claude Code Harness and GPT-5.4-mini on ALFWorld, WebShop and ScienceWorld.

Method	ALFWorld				WebShop		ScienceWorld			
	Seen		*Unseen*		*Evaluation*		*Seen*		*Unseen*	
	$R\uparrow$	$S\downarrow$	$R\uparrow$	$S\downarrow$	$R\uparrow$	$S\downarrow$	$R\uparrow$	$S\downarrow$	$R\uparrow$	$S\downarrow$
SkillsWiki	**88.57**	15.51	**89.55**	15.98	**17.13**	**26.14**	**85.09**	13.87	**85.37**	**15.06**
w/o taxonomy	81.43	**14.99**	83.58	**15.26**	7.45	27.19	83.82	14.84	81.22	15.61
w/o relations	76.43	17.11	85.07	16.38	6.73	27.92	82.06	14.61	80.63	16.50
w/o packages	75.71	18.45	82.09	17.31	10.35	27.18	83.99	**13.32**	82.41	15.50

cating that experience-grounded reuse is the most critical component, followed by relations, which allow the agent to assemble appropriate compositions rather than isolated skills. Removing the taxonomy also hurts, though less severely, and some ablated variants use fewer steps at a substantial cost in success rate. All three layers are necessary and jointly produce the observed improvement.

6.3 Retrieval Effectiveness Analysis



(a) Reward (b) Task progress (c) Skill reward contribution

Figure 3: Effectiveness analysis of retrieved skills

Figure 3 reports three complementary views of retrieved-skill quality, all measured where both methods successfully retrieve skills, so the comparison reflects skill quality rather than retrieval success. Reward measures task performance, task progress measures the fraction of episodes with positive reward, and skill reward contribution multiplies reward by an LLM judgment of whether the retrieved skills actually helped, separating useful retrieval from retrieval that merely co-occurs with success. SkillsWiki leads on all three, with the largest gap where several skills must be coordinated. The contribution metric is the most informative, since it confirms that SkillsWiki's retrieved skills are judged helpful, not merely associated with high reward. SkillNet retrieves comparable numbers of skills but converts them into success less often, consistent with a flat skill space lacking relational and experiential structure. The benefit of SkillsWiki therefore comes from the usefulness of retrieved skills rather than from retrieval quantity.

6.4 Wiki Construction Overhead

Table 3: Wiki construction statistics.

Statistical Item	ALFWorld	WebShop	ScienceWorld
Skill Taxonomy Construction Time (s)	111	118	202
Relation Graph Construction Time (s)	273	1119	416
Skill Package Construction Time (s)	144	327	140

Table 3 reports the one-time overhead of constructing a SkillsWiki on each benchmark. Relation discovery is consistently the dominant cost, as it entails agent-driven exploration of the wiki together

with validation of each candidate skill, whereas taxonomy and package construction remain comparatively inexpensive. The elevated relation cost on WebShop, at 1119 seconds, is consistent with its larger and more interdependent skill inventory. The aggregate overhead ranges from roughly 9 to 26 minutes per benchmark and is incurred once rather than at inference time. Because the resulting wiki is reused across all evaluation episodes and supports incremental updates, this cost is amortized over the benchmark and does not contribute to per-task latency.

6.5 Case Study

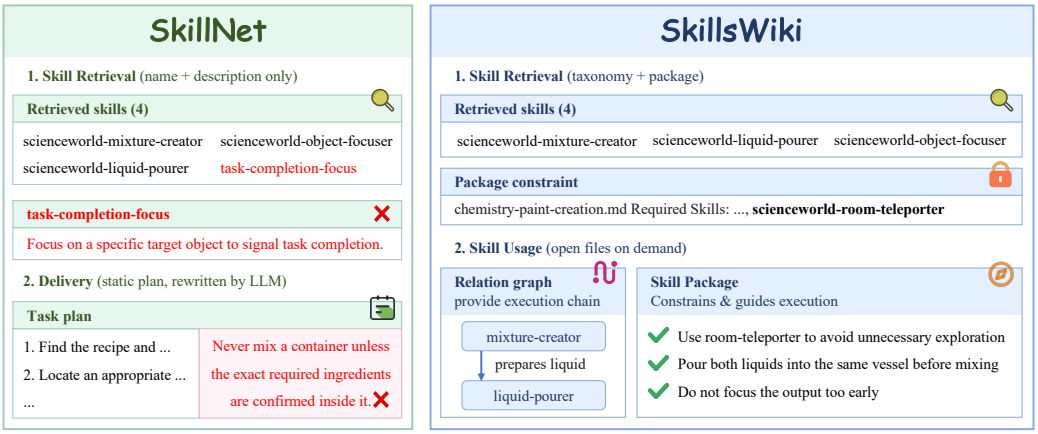


Figure 4: **Case study.** SkillsWiki retrieves the skills that match the task through taxonomy, relations, and packages, while SkillNet retrieves a conflicting set and delivers them as a rigid plan.

We examine a ScienceWorld paint-mixing task that requires focusing on an intermediate secondary-colour paint before focusing on the final violet paint. Both methods draw on a byte-identical 61-skill library, yet they retrieve different skills. As shown in Figure 4, SkillsWiki returns exactly the four skills required by the package chemistry-paint-creation, including room-teleporter, whereas SkillNet returns the same three core skills plus task-completion-focus and omits the teleporter. That substitution is decisive. task-completion-focus is defined only for signalling final completion and explicitly forbids intermediate use, directly contradicting the task, while room-teleporter is precisely what the episode needs to avoid redundant exploration. Delivery also diverges. SkillNet rewrites skill bodies into a static plan that promotes a reference suggestion into a blocking precondition and discards verbatim action examples, so its tool-free agent exhausts all 30 steps searching for a recipe that does not exist, performing zero pour, mix, or focus actions. SkillsWiki keeps skills as addressable files, allowing the agent to verify that no recipe exists, follow the relation from mixture-creator to liquid-pourer, and reproduce the documented pour pattern. It scores 100 in 15 steps against SkillNet's 13 in 30 steps. The taxonomy locates the right skills, the relation graph supplies the pour-before-mix chain, and the package constrains and guides execution, jointly turning retrieval into successful task completion.

7 Conclusion

We present SkillsWiki, a hierarchical wiki-structured skill repository for LLM agents. SkillsWiki organizes skills into three complementary layers, a taxonomy for coarse-to-fine retrieval, a relation graph for navigating related skills, and a package library for reusing proven compositions, all maintained as a lightweight and human-readable wiki. This design addresses the flat, isolated, and static nature of prevailing skill management, and supports autonomous relation discovery as well as experience-grounded reuse. Experiments on ALFWorld, WebShop, and ScienceWorld show consistent gains in average reward and execution efficiency over strong baselines. SkillsWiki thus offers a scalable and extensible substrate for organizing, retrieving, and reusing agent skills.

AI USE STATEMENT

In this work, we used generative AI tools for writing polishing, experimental evaluation and analysis. We have not used generative AI tools for idea generation and literature retrieval, and synthetic dataset generation are not applicable to this work. We have reviewed all AI-assisted work. We reviewed the LLM-polished paper content, as well as the experimental evaluation and statistical analysis. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility, we provide comprehensive descriptions of our method and experimental setup. The overall structure of SkillsWiki is detailed in Section 3, and its construction and retrieval procedures are specified in Section 4. The experimental settings, including datasets, baselines, and evaluation metrics, are described in Section 5 and Appendix A. All prompts used in the experiments are provided in Appendix D. We will release the source code upon publication.

REFERENCES

Aili Chen, Aonian Li, Baichuan Zhou, Bangwei Gong, Binyang Jiang, Boji Dan, Changhao Zhang, Changqing Yu, Chao Wang, Cheng Ma, et al. The minimax-m2 series: Mini activations unleashing max real-world intelligence. *arXiv preprint arXiv:2605.26494*, 2026.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready ai agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*, 2025.

GLM-5-Team, :, Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, Chenzheng Zhu, Congfeng Yin, Cunxiang Wang, Gengzheng Pan, Hao Zeng, Haoke Zhang, Haoran Wang, Huilong Chen, Jiajie Zhang, Jian Jiao, Jiaqi Guo, Jingsen Wang, Jingzhao Du, Jinzhu Wu, Kedong Wang, Lei Li, Lin Fan, Lucen Zhong, Mingdao Liu, Mingming Zhao, Pengfan Du, Qian Dong, Rui Lu, Shuang-Li, Shulin Cao, Song Liu, Ting Jiang, Xiaodong Chen, Xiaohan Zhang, Xuancheng Huang, Xuezhen Dong, Yabo Xu, Yao Wei, Yifan An, Yilin Niu, Yitong Zhu, Yuanhao Wen, Yukuo Cen, Yushi Bai, Zhongpei Qiao, Zihan Wang, Zikang Wang, Zilin Zhu, Ziqiang Liu, Zixuan Li, Bojie Wang, Bosi Wen, Can Huang, Changpeng Cai, Chao Yu, Chen Li, Chengwei Hu, Chenhui Zhang, Dan Zhang, Daoyan Lin, Dayong Yang, Di Wang, Ding Ai, Erle Zhu, Fangzhou Yi, Feiyu Chen, Guohong Wen, Hailong Sun, Haisha Zhao, Haiyi Hu, Hanchen Zhang, Hanrui Liu, Hanyu Zhang, Hao Peng, Hao Tai, Haobo Zhang, He Liu, Hongwei Wang, Hongxi Yan, Hongyu Ge, Huan Liu, Huanpeng Chu, Jia'ni Zhao, Jiachen Wang, Jiajing Zhao, Jiamin Ren, Jiapeng Wang, Jiaxin Zhang, Jiayi Gui, Jiayue Zhao, Jijie Li, Jing An, Jing Li, Jingwei Yuan, Jinhua Du, Jinxin Liu, Junkai Zhi, Junwen Duan, Kaiyue Zhou, Kangjian Wei, Ke Wang, Keyun Luo, Laiqiang Zhang, Leigang Sha, Liang Xu, Lindong Wu, Lintao Ding, Lu Chen, Minghao Li, Nianyi Lin, Pan Ta, Qiang Zou, Rongjun Song, Ruiqi Yang, Shangqing Tu, Shangtong Yang, Shaoxiang Wu, Shengyan Zhang, Shijie Li, Shuang Li, Shuyi Fan, Wei Qin, Wei Tian, Weining Zhang, Wenbo Yu, Wenjie Liang, Xiang Kuang, Xiangmeng Cheng, Xiangyang Li, Xiaoquan Yan, Xiaowei Hu, Xiaoying Ling, Xing Fan, Xingye Xia, Xinyuan Zhang, Xinze Zhang, Xirui Pan, Xu Zou, Xunkai Zhang, Yadi Liu, Yandong Wu, Yanfu Li, Yidong Wang, Yifan Zhu, Yijun Tan, Yilin Zhou, Yiming Pan, Ying Zhang, Yinpei Su, Yipeng Geng, Yong Yan, Yonglin Tan, Yuean Bi, Yuhan Shen, Yuhao Yang, Yujiang Li, Yunan Liu, Yunqing Wang, Yuntao Li, Yurong Wu, Yutao Zhang, Yuxi Duan, Yuxuan Zhang, Zezhen Liu, Zhengtao Jiang, Zhenhe Yan, Zheyu Zhang, Zhixiang Wei, Zhuo Chen, Zhuoer Feng, Zijun Yao, Ziwei Chai, Ziyuan Wang, Zuzhou Zhang, Bin Xu, Minlie Huang, Hongning Wang, Juanzi Li, Yuxiao Dong, and Jie Tang. Glm-5: from vibe coding to agentic engineering, 2026. URL https://arxiv.org/abs/2602.15763.

Zhongxin Guo, Danrui Qi, Hanwen Gu, Peng Cheng, and Yongqiang Xiong. Skill-disco: Distilling and compiling agent traces into reusable procedural skills. *arXiv preprint arXiv:2606.26669*, 2026.

Yu Hao, Jinxuan Cai, Qi Zhang, Yawen Li, Zhiqiang Zhang, Chuan Shi, and Cheng Yang. Hiskill: Empowering llm agents with hierarchical skill graphs. *arXiv preprint arXiv:2607.25853*, 2026.

Shiqi He, Yue Cui, Feijie Wu, Xinyu Ma, Jiaheng Lu, Yaliang Li, Bolin Ding, and Mosharaf Chowdhury. Beyond domains: Reusing web skills via transferable interaction patterns. *arXiv preprint arXiv:2606.17645*, 2026.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Steven Yau, Zijuan Lin, Liyang Zhou, et al. Metagpt: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations*, volume 2024, pp. 23247–23275, 2024.

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*, 2026.

Yubo Li. Dynamic agent skills: A lifecycle survey and taxonomy of evolving skill libraries. *arXiv preprint arXiv:2607.10113*, 2026.

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

Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. Gorilla: Large language model connected with massive apis. *Advances in Neural Information Processing Systems*, 37:126544–126565, 2024.

Xue Bin Peng, Michael Chang, Grace Zhang, Pieter Abbeel, and Sergey Levine. Mcp: Learning composable hierarchical control with multiplicative compositional policies. *Advances in neural information processing systems*, 32, 2019.

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

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. *Advances in Neural Information Processing Systems*, 36:38154–38180, 2023.

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 Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*, 2020.

Weihang Su, Jianming Long, Qingyao Ai, Qiaozhi He, Yichen Tang, Changyue Wang, Yiteng Tu, Yingbo Wang, and Yiqun Liu. Skill retrieval augmentation for agentic ai. *arXiv preprint arXiv:2604.24594*, 2026.

Wenbiao Tao, Xinyuan Li, Yunshi Lan, and Weining Qian. Tagrag: tag-guided hierarchical knowledge graph retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 6434–6456, 2026.

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.

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

Bin Wu, Edgar Meij, and Emine Yilmaz. Beyond static toolsets: Self-evolving llm tool agents via continual documentation adaptation. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 21519–21539, 2026a.

Jiangrong Wu, Yuhong Nan, Yixi Lin, Huaijin Wang, Yuming Xiao, Shuai Wang, and Zibin Zheng. Skillscope: Toward fine-grained least-privilege enforcement for agent skills. *arXiv preprint arXiv:2605.05868*, 2026b.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757, 2022a.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022b.

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.

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.

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.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning acting and planning in language models. *arXiv preprint arXiv:2310.04406*, 2023.

A Detailed Experimental Settings

A.1 Experimental Environment

Table 4 lists the relevant experimental environment information, including hardware and software.

Table 4: Experimental environment.

	Configuration
CPU	INTEL(R) XEON(R) GOLD 6530
RAM	128GB
Operating System	Ubuntu 22.04 LTS
Python	3.12.13

Table 5 shows the parameters for the execution of the Benckmark task, the configuration of Harness, and the calls to the LLM (Large Language Model).

Table 5: Hyperparameter settings

Hyperparameter	**Value**	**Description**
Benchmark execution		
Max steps	30	Max environment actions
Max turns	120	Max conversation turns
Max thinking tokens	8000	Max thinking tokens
Server timeout	300	HTTP timeout (seconds)
Harness settings		
Harness	claude-code	Execution harness
Permission mode	acceptEdits	Permission mode
Allowed tools	Read/Glob/Grep/Bash/Write /episode_observe/episode_step/episode_status	Tools allowed for harness
LLM call		
Temperature	1	LLM generation temperature
LLM timeout	120.0 s	Chat and embedding
Retry num	3	Number of LLM retries

A.2 Dataset Details

We conduct experiments on three datasets, namely ALFWorld, WebShop and ScienceWorld.

ALFWorld, WebShop, and ScienceWorld are three widely used interactive text environment benchmarks for language agents, covering embodied household tasks, online shopping, and scientific experimentation respectively.

ALFWorld is a text based embodied environment built upon ALFRED, converting visual embodied tasks into pure text interaction. The environment contains six types of household tasks such as picking, cleaning, heating, and cooling. The training set includes 3,553 task instances, and the evaluation set is divided into in distribution and out of distribution portions. Agents must complete goals over multiple turns based on textual observations and a list of available actions. The core challenges lie in long horizon state tracking and efficient exploration under sparse rewards.

WebShop simulates a realistic e commerce website with 1.18 million real product entries and 12,087 crowdsourced text instructions. Given a product requirement instruction specifying attributes such as color, price, or size, an agent must navigate multiple page types including search pages, result lists, product detail pages, and option selection pages before finally choosing and purchasing a product. The main difficulties include compositional instruction understanding, query rewriting, noisy web text processing, and strategic exploration. Evaluation metrics are purchase success rate and match reward score.

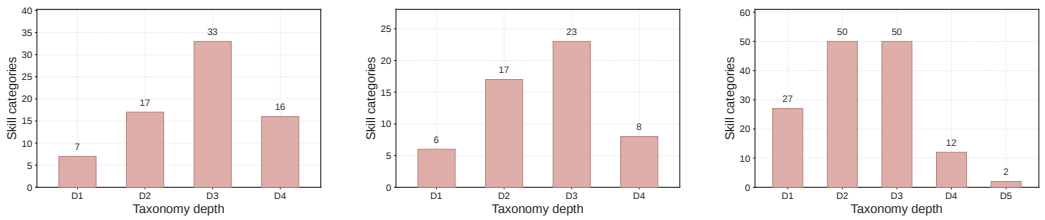
ScienceWorld is an interactive text environment aligned with elementary school science curricula, covering ten task themes such as states of matter, measurement, electricity, biology, and chemistry. Its purpose is to test scientific reasoning in an interactive setting rather than through static question answering. A key characteristic of this benchmark is that the evaluation set contains unseen task variations, for example learning to boil water during training while being asked to boil lead during testing, which is designed to assess generalization ability.

Table 6: Dataset details.

	ALFWorld	WebShop	ScienceWorld
Domain	Embodied Intelligence	Web interaction	Science education
Actions	go to, take, heat, put, etc.	search, choose, etc.	use, move, focus, pour, etc.
Train set	3553	11887	1483
Dev set	140	/	194
Test set	134	200	211

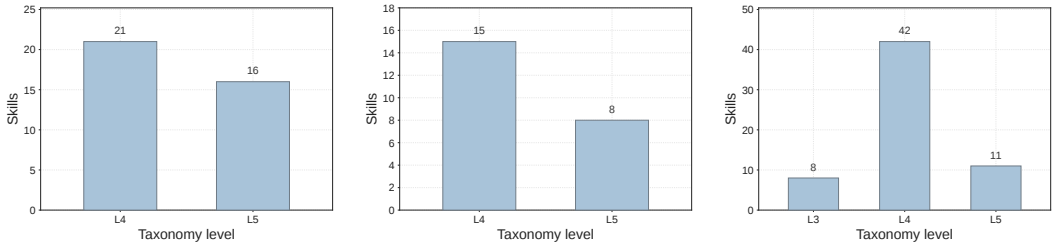
B Experimental Results and Analysis

B.1 Skill Distribution Analysis



(a) ALFWorld (b) WebShop (c) ScienceWorld

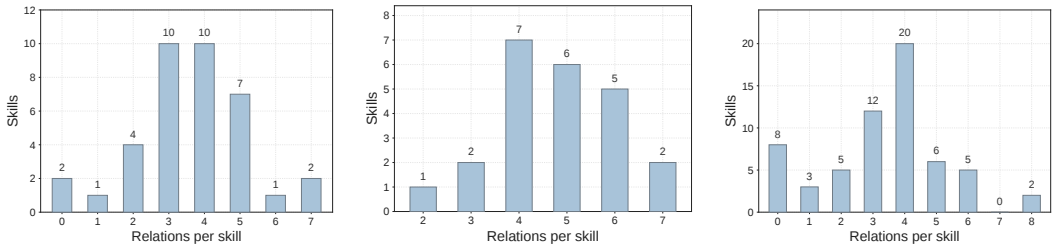
Figure 5: Skill taxonomy category distribution



(a) ALFWorld (b) WebShop (c) ScienceWorld

Figure 6: Skill taxonomy distribution

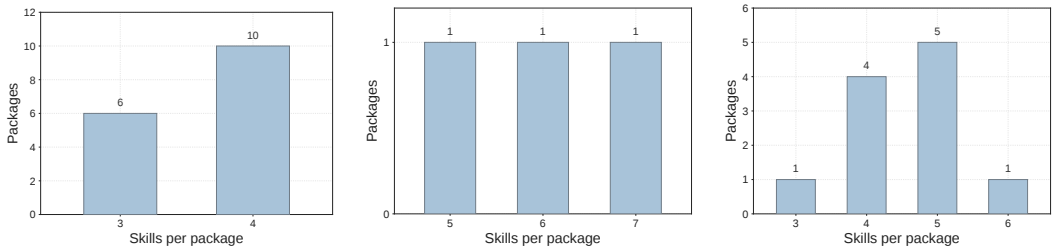
Skill Taxonomy Distribution Analysis Figures 5 and 6 characterize the induced taxonomy from two complementary angles. The first reports how many distinct categories exist at each depth, and the second reports how many skills fall at each depth. Across all three benchmarks, the category counts grow in breadth before they grow in specificity, and the skill counts follow the same tendency toward depth. Two observations follow. First, the taxonomy is genuinely nested, since both categories and skills are spread across several levels rather than collapsed into a flat label set. Second, the number of categories grows faster than the number of skills at shallow depths, which means the classifier prefers to introduce intermediate categories instead of overloading a few top-level nodes. This is precisely the structure that makes coarse-to-fine retrieval effective, because the agent can prune large parts of the skill space at each level while still reaching specific leaf skills.



(a) ALFWorld (b) WebShop (c) ScienceWorld

Figure 7: Skill relation distribution

Skill Relation Distribution Analysis Figure 7 reports how many relations each skill has. The distributions are far from degenerate on all three benchmarks, which confirms that the discovered graph is genuinely populated rather than a few isolated hubs. Only a small minority of skills remain isolated, so relation discovery succeeds for nearly the whole skill set rather than for a handful of popular skills. Moreover, the graph becomes denser as the underlying skill inventory grows larger and more interdependent, since domains whose workflows involve more prerequisites and companion skills naturally yield a heavier tail of high-degree skills. The graph therefore adapts its connectivity to the complexity of the domain instead of imposing a uniform degree.



(a) ALFWorld (b) WebShop (c) ScienceWorld

Figure 8: Skill package distribution

Skill Package Distribution Analysis Figure 8 shows how many skills each distilled package contains. Package sizes stay within a moderate range on every benchmark rather than growing with the number of observed episodes, which indicates that the abstraction and merging step collapses near-duplicate compositions into compact reusable units instead of accumulating every observed sequence. The variation in package size across benchmarks also tracks the underlying skill complexity, since domains whose workflows require more cooperating skills naturally yield longer compositions while simpler task spaces favor shorter ones. The experience layer therefore adapts to the domain rather than imposing a fixed composition length.

C Application Scenarios

SkillsWiki is designed as a general substrate for skill organization and retrieval, and its value becomes most visible when the skill space is large, heterogeneous, and shared across environments. We illustrate this with two scenarios. The first shows how SkillsWiki supports targeted retrieval over a large scientific skill collection, and the second shows how the same structure coordinates a shared cloud-side wiki with a local workspace wiki.

C.1 Scientific Discovery Skills Retrieval

Figure 9 shows a SkillsWiki instance built over 660 scientific skills, accessed through a web interface that exposes three views of the same wiki. The left panel presents the wiki folder tree, where skills are nested under taxonomy categories such as bioinformatics and genomics and each node reports



Figure 9: A SkillsWiki instance built over 660 scientific skills.

how many skills it contains. The center panel shows the knowledge graph of the wiki, which displays the overall skill structure in the default state and switches to the relation graph among retrieved skills once a query is issued. The right panel shows the detail of a selected skill, including its path, description, tools, workflow, test case, and the reason it was retrieved, so that a user can inspect why the system surfaced it.

Retrieval supports three modes. The keyword mode performs purely local BM25 scoring over skill metadata with a phrase boost, which is fast, deterministic, and requires no model or network access, though it is sensitive to surface wording. The harness mode delegates retrieval to an agent that browses the taxonomy readmes, metadata, and relation and package references before returning structured results, which captures semantic and structural relevance at higher latency. The hybrid mode first selects a small set of seed skills by keyword and then runs a shallow harness on each seed within its local relation neighborhood, combining the two result sets to balance cost and accuracy.

C.2 Cloud-local Collaborative Skill Organization

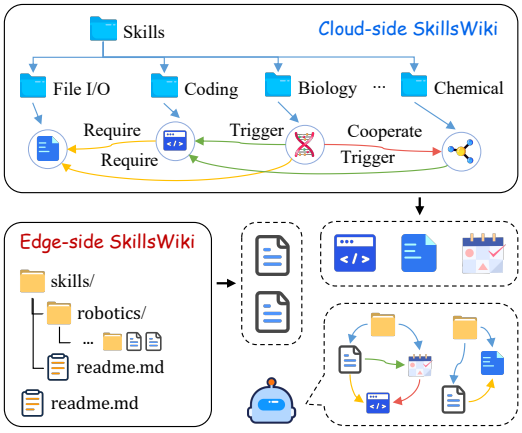


Figure 10: Cloud-local SkillsWiki Collaboration.

Figure 10 illustrates how SkillsWiki couples a cloud-side wiki with an local-side workspace wiki. The cloud-side SkillsWiki is a published, read-only repository that hosts a large and broadly shared skill collection organized under a common taxonomy, together with the relations among those skills. The local-side SkillsWiki is a lightweight workspace that holds only the skills currently materialized

for the task at hand, arranged in the same taxonomy so that local and remote structures remain compatible.

The two sides cooperate through agent-driven discovery. When the agent retrieves skills locally, it inspects their relation records and follows the outgoing edges that point to the cloud side. Rather than downloading the whole remote taxonomy, it selects only the leaf skill folders that are relevant to the local skills already in use, and imports them into the local-side workspace while preserving their taxonomy paths. Each imported skill keeps a provenance marker that distinguishes it from native ones, and the newly imported skills may in turn expose further relations, so discovery and import proceed incrementally. This design keeps the local footprint small, allows the cloud wiki to grow without bound, and lets the agent compose general cloud skills with specialized local ones under a single wiki abstraction.

D PROMPTS

We present the prompts used for wiki construction and task execution below.

Prompt of skill taxonomy construction

You build concise skill taxonomy paths. Return only XML. The first two nodes are fixed and must be skills and the provided top-level category. Add 1 to 4 broad-to-specific lowercase English intermediate category nodes when useful. The final node must be the skill name. Do not add explanations.

Classify this skill into a multi-level taxonomy.

Fixed prefix:
<node>skills</node>
<node>{category}</node>

Skill name: {skill.name}
Skill description: {skill.description}

SKILL.md front matter:
{front_matter}

Return exactly:
```
<taxonomy>
  <path>
    <node>skills</node>
    <node>{category}</node>
    <node>...</node>
    <node>{skill.name}</node>
  </path>
</taxonomy>
```

Prompt of skill category summary generation

You maintain README files for a skill taxonomy wiki. The README is used for retrieval and filtering, so it must be short, high-signal, and conceptual. Summarize the current taxonomy level: what problem types it covers, how its direct child categories differ, and what capabilities its skills offer. Do not copy raw skill descriptions, usage commands, URLs, metadata fields, full taxonomy paths, or complete directory structures. Merge repeated ideas instead of appending duplicates. Do not invent unrelated skills. Return only the final Markdown README. Never include hidden reasoning, chain-of-thought, <think> blocks, planning notes, or commentary.

Current taxonomy chain:
{current_chain}

Existing readme.md:
{existing_readme}

Direct child evidence:
{child_context}

Rewrite the complete readme.md for the current taxonomy directory.
If Existing readme.md is not empty, treat it as the current accumulated summary
for earlier child evidence and incrementally merge the new Direct child evidence
into it. The output must remain a complete replacement README that covers both
the existing summary and the new evidence.
Return only the final Markdown content. Do not include analysis, reasoning, XML,
JSON, code fences, <think> blocks, or commentary before/after the Markdown.

The first line must be the Markdown H1 title for the current taxonomy directory.

Required structure:
<short taxonomy title>

Abstract
2-5 short sentences summarizing this directory's scope and the types of user problems it can solve.

Contents Overview
Summarize direct child categories or direct skill groups by capability. Do not list every child when there are many.

Skill Capability
Describe capability clusters in short sentences rather than full skill descriptions.

Prompt of skill relation graph construction

You are a SkillsWiki relation graph builder. Return only strict JSON.

Source skill:
- name: {skill.name}
- path: {skill.rel_path}
- taxonomy_chain: {chain_text}
- description: {skill.description}

Target SkillsWiki:
- root: {args.SkillsWiki_path}
- local taxonomy root: {taxonomy_root}

Task:
- Find closely related existing SkillsWiki skills for the source skill.
- Use paths exactly as they appear relative to the SkillsWiki root, for example "wiki/..." or "skills/...".
- Inspect the local SkillsWiki taxonomy under {taxonomy_root}.
- Use workspace guidance files only as SkillsWiki management policy.
- Use the current prompt as the concrete task instruction.
- Do not invent paths.

Output JSON only:

```
{
  "relations": [
    {
      "path": "skills/...",
      "name": "skill-name",
      "reason": "short English reason in one sentence"
    }
  ]
}
```

Prompt of skill package construction

Build skill packages for this workspace from the existing benchmark trainset results.

Workspace: {workspace}
Benchmark result directory: {result_dir}
Dataset: {dataset}

Split: {split}

Follow the Build Skill Package Workflow in AGENTS.md and CLAUDE.md.

Required outputs:
- Write or update <workspace>/skill_packages/index.md.
- Write or update package detail files under <workspace>/skill_packages/<package-name>.md.
- Write compact reverse package references to relevant leaf skill folders as <workspace>/skills/**/packages.md.

Use every idx_*.json file in the benchmark result directory as the source evidence, but do not read the raw full JSON files one by one with the Read tool. The result directory can be large and each file contains full message/trajectory history. First use shell or Python commands to mechanically scan all idx_*.json files and create compact evidence that preserves only the fields needed for package construction: task/query text, reward, task_done/ok status, steps, retrieved skill paths, relevant skill paths, concise action/observation milestones, and concise failure notes. Use successful trajectories for package procedures and required skills; use failed or low-reward trajectories only for Failure Guards. Then read the compact evidence and write the package files. If scratch files are needed, place them under <workspace>/cache/ and remove them before finishing. If package files already exist, treat them as current package memory and update/merge them instead of duplicating near-identical packages.

Do not rerun benchmark tasks. Do not call benchmark environment endpoints. Do not modify SKILL.md, metadata.json, taxonomy.xml, readme.md, relations.json, or relations.md.

When finished, briefly report which package files were written and how many per-skill packages.md references were updated.

Prompt of ALFWorld task execution

Interact with a household to solve a task. Imagine you are an intelligent agent in a household environment and your target is to perform actions to complete the task goal. At the beginning of your interactions, you will be given the detailed description of the current environment and your goal to accomplish.
For each of your turn, you will be given the observation of the last turn. You should first think about the current condition and plan for your future actions, and then output your action in this turn. Your output must strictly follow the format: "Thought: your thoughts. InAction: your next action".

The available actions are:
1. go to {recep}
2. take {obj} from {recep}
3. move {obj} to {recep}
4. open {recep}
5. close {recep}
6. use {obj}
7. clean {obj} with {recep}
8. heat {obj} with {recep}
9. cool {obj} with {recep}
where {obj} and {recep} correspond to objects and receptacles.
After your each turn, the environment will give you immediate feedback based on which you plan your next few steps. if the environment output "Nothing happened", that means the previous action is invalid and you should try more options

Your response should use the following format:

Thought: <your thoughts>
Action: <your next action>

Prompt of ScienceWorld task execution

You are a helpful assistant to do some scientific experiment in an environment.
In the environment, there are several rooms: kitchen, foundry, workshop, bathroom, outside, living room, bedroom, greenhouse, art studio, hallway

You should explore the environment and find the items you need to complete the experiment.

You can teleport to any room in one step.
All containers in the environment have already been opened, you can directly get items from the containers.
The available actions are:

open OBJ: open a container
close OBJ: close a container
activate OBJ: activate a device
deactivate OBJ: deactivate a device
connect OBJ to OBJ: connect electrical components
disconnect OBJ: disconnect electrical components
use OBJ [on OBJ]: use a device/item
Look around: describe the current room
examine OBJ: describe an object in detail
look at OBJ: describe a container's contents
read OBJ: read a note or book
move OBJ to OBJ: move an object to a container
pick up OBJ: move an object to the inventory
pour OBJ into OBJ: pour a liquid into a container
mix OBJ: chemically mix a container
teleport to LOC: teleport to a specific room
focus on OBJ: signal intent on a task object
wait: task no action for 10 steps
wait1: task no action for a step

Your response should use the following format:

Thought: <your thoughts>
Action: <your next action>

Prompt of WebShop task execution

You are web shopping.
I will give you instructions about what to do.
You have to follow the instructions.
Every round I will give you an observation and a list of available actions, you have to respond an action based on the state and instruction.
You can use search action if search is available.
You can click one of the buttons in clickables.
An action should be of the following structure:
search [keywords]
click[value]
If the action is not valid, perform nothing.
Keywords in search are up to you, but the value in click must be a value in the list of available actions.
Remember that your keywords in search should be carefully designed.

Your response should use the following format:

Thought: <your thoughts>
Action: <your next action>