SKILLNET-GYM: A DYNAMIC BENCHMARK FOR COMPOSITIONAL SKILL LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model (LLM) agents increasingly rely on skills to solve complex, long-horizon tasks. However, existing benchmarks for agent skills are mostly static, manually curated snapshots, making them poorly suited to the continuously evolving nature of real-world skill ecosystems. To address this gap, we introduce SkillNet-Gym, a dynamic benchmark for evaluating compositional skill learning in LLM agents. Grounded in real-world sources, SkillNet-Gym automatically builds a continuously evolving heterogeneous SkillNet, and uses it to synthesize domain-diverse, compositionally complex tasks. These tasks enable unified evaluation for skill construction and skill composition. Evaluating eight frontier models with two widely used agent harnesses reveals that directly compressing documents into skills can cause information loss and agents remain weak at retrieving and orchestrating skills in the wild. These findings show that effective skill use is not a one-shot augmentation mechanism, but a compositional pipeline of interdependent competencies. SkillNet-Gym provides a dynamic testbed for developing, diagnosing, and improving evolving skill centric agents.

1 INTRODUCTION

With the deep integration of large language models (LLMs) (OpenAI, 2025b; GLM-5-Team et al., 2026; DeepSeek-AI, 2026) and harnesses (Anthropic, 2025; Google, 2025; OpenAI, 2025a; Lou et al., 2026; Lee et al., 2026), agents have demonstrated remarkable advantages in executing long-horizon tasks (Li et al., 2025; Ye et al., 2026; Zhang et al., 2026d) and interacting with real-world environments (Bandi et al., 2026; Luo et al., 2025; Kilo Code, 2026; Merrill et al., 2026) to solve complex, open-ended tasks. A key mechanism underlying these capabilities is the use of **skills** (Anthropic, 2025): structured resources that encapsulate reusable workflows, executable modules, and supporting references. Beyond merely consuming existing skills, effective agents must also support two higher-level capabilities: *(1) Skill construction* (Chen et al., 2026b; Wang et al., 2026a; Ni et al., 2026; Yang et al., 2026) refers to distilling raw resources, such as documents, codebases, or past trajectories, into reusable skills; *(2) Skill Composition* (Zheng et al., 2026; Cho et al., 2026; Li et al., 2026a) refers to retrieving multiple relevant skills and organizing them into coherent workflows with correct dependency handoffs. Together, these capabilities are critical for improving agent performance, adaptability, and robustness on complex tasks.

Motivated by this potential, a growing body of work has focused on developing benchmarks for agent skills. However, most existing benchmarks are still built as static, manually curated snapshots (Li et al., 2026b; Han et al., 2026; Zhong et al., 2026; Liu et al., 2026c). This design increasingly conflicts with the nature of real-world skill ecosystems, which evolve continuously over time. As the skill ecosystem evolves, static benchmarks quickly drift away from original data distributions they are designed to represent, fundamentally compromising the validity of evaluation. This limitation raises a fundamental question: **How should we benchmark agent skills when the skill ecosystem itself is continuously evolving?**

To answer this question, we introduce **SkillNet-Gym**, a dynamic benchmark for evaluating compositional skill learning in LLM agents (MIT License). As shown in Figure 1, SkillNet-Gym aligns benchmark construction with the real world by organizing community-contributed skills and related Internet artifacts into a continuously evolving heterogeneous **SkillNet**. This design ensures that the benchmark is derived from authentic artifacts, reflects the current state of the ecosystem, and remains

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.

Benchmark	Benchmark Construction			Benchmark Scope			Evaluation Focus	
	Real-World Source	Automatic Construction	Dynamic Extensibility	Domain Coverage	Skill Pool	Task Numbers	Skill Construction	Skill Composition
SkillsBench Li et al. (2026b)	✓	✗	✗	✓	✗	87	✗	✗
SWE-Skills-Bench Han et al. (2026)	✓	✗	✗	✗	✗	565	✗	✗
SkillLearnBench Zhong et al. (2026)	✓	✗	✗	✓	✗	20	✓	✗
WildSkillBench Liu et al. (2026c)	✓	✗	✗	✓	✓	87	✓	✓
SkillRouter (Zheng et al., 2026)	✓	✓	✓	✓	✓	75	✗	✓
SkillRet (Cho et al., 2026)	✓	✓	✓	✓	✓	4997	✗	✓
AgentSkillOS Li et al. (2026a)	✓	✗	✗	✓	✓	30	✗	✓
SkillGenBench Zhou et al. (2026b)	✓	✓	✓	✓	✗	187	✓	✗
OpenSkillEval Ying et al. (2026)	✓	✓	✓	✓	✗	600	✗	✗
SkillNet-Gym (ours)	✓	✓	✓	✓	✓	159	✓	✓

Table 1: Comparison of existing agentic skills benchmarks across three dimensions: **Benchmark Construction**, **Benchmark Scope** and **Evaluation Focus**. "✓" indicates fully coverage, "✓" indicates partially coverage, and "✗" indicates not coverage. Detailed description for each trait are provided in Appendix B.

extensible as new skills emerge. Even when existing tasks or skills become outdated, SkillNet-Gym can continuously synthesize new benchmark tasks, allowing the benchmark to evolve together with the skill ecosystem.

Built on this foundation, SkillNet-Gym employs an automated benchmark construction pipeline that transforms community skills into executable evaluation tasks. The pipeline continuously collects and normalizes skills, infers inter-skill relations, constructs a heterogeneous skill graph, samples compositional subgraphs, and synthesizes task instructions, reference solutions, and verifiers under multi-stage quality control. In total, we construct 159 tasks spanning diverse domains across multiple disciplines. Over these tasks, SkillNet-Gym supports a unified evaluation, including **Skill Construction** and **Skill Composition**. Under this common protocol, we evaluate eight frontier models together with two widely used agent harnesses, Claude Code (Anthropic, 2025) and Codex (OpenAI, 2025a), enabling controlled comparison across different stages of skill use. Our empirical results show that skill utility is highly conditional: skills are most beneficial for hard tasks, while one-shot skill construction may lead to information loss and human-curated skills may introduce knowledge bias or override task instructions. In realistic wild repositories, agents struggle to retrieve complete skill sets and organize them into correct dependency workflows. These findings suggest that **agents should not simply rely on fixed skills as static resources. Instead, effective skill use requires agents to compose skills before applying them, while continuously adapting these skills based on their own execution behavior and the evolving external ecosystem.** SkillNet-Gym can also provide the dynamic feedback loop needed for such self-adaptive skill learning.

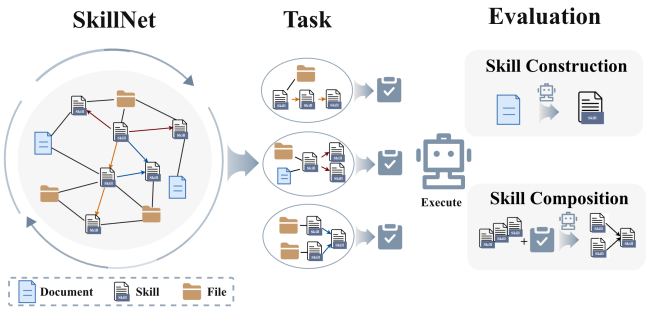


Figure 1: Starting from real-world resources, SkillNet-Gym continuously collects and organizes these artifacts into a heterogeneous SkillNet. Built upon this foundation, the system automatically constructs a dynamically updated, quality-controlled benchmark. The resulting benchmark enables unified evaluation across **Skill Construction** and **Skill Composition**.

Our contributions are threefold:

- We introduce SkillNet-Gym, a dynamically updated benchmark for evaluating agents' compositional skill learning.
- We develop an automated, community-grounded pipeline that constructs an evolving SkillNet from real-world skills and synthesizes executable and domain-diverse benchmark tasks.

- We present a unified empirical analysis of skill construction and skill composition across frontier models and agent harnesses, revealing the conditions under which skills help, fail, and should be adapted.

2 PRELIMINARIES

2.1 TASK SOLVING WITH AGENT SKILLS

We consider an interactive task setting with optional skill access, where each task is defined as $x = (\mathcal{I}, \mathcal{E}, \mathcal{M})$, where $\mathcal{I}$ is the instruction, $\mathcal{E}$ is the execution environment, and $\mathcal{M}$ is the evaluation protocol. Given x, the agent produces a trajectory $\tau = (o_1, a_1, \ldots, o_T, a_T)$ of observations and actions, and succeeds if the final output passes $\mathcal{M}$.

A *skill* is a reusable procedural artifact that helps solve a class of subtasks, such as a tool-use recipe, executable module, or structured document. We represent a skill as $s = (m_s, r_s, c_s)$, where m_s is the core procedural content of the skill, r_s denotes optional auxiliary resources (e.g., templates, examples, or executable snippets), and c_s denotes its applicability condition for retrieval and use. Given a task x, the agent may use a task-specific skill set $\mathcal{S}_x = \{s_1, \ldots, s_k\}$, yielding $\tau \sim \pi_\theta(\cdot \mid x, \mathcal{S}_x)$, where π_θ is the agent policy. The purpose of skill augmentation is that, compared with solving tasks without external skills, conditioning on relevant skills should improve both execution efficiency and task performance.

2.2 THE LIFECYCLE OF AGENT SKILLS

We view skill use in LLM agents as a lifecycle of progressively richer capabilities that connect skill creation, skill selection, workflow organization, and task execution. A realistic benchmark should therefore distinguish whether an agent can solve a task without external procedural support, benefit from given skills, construct reusable skills from upstream resources, and compose multiple skills under noisy repository settings. SkillNet-Gym instantiates this view with four evaluation settings: *(1) No Skill:* The agent solves tasks using only its parametric knowledge and the task context, without any external skills. *(2) Skill Efficacy:* The agent is provided with official skills before task solving, e.g., from repositories such as ClawHub and SkillHub. This setting evaluates whether externally supplied skills can be correctly understood and applied during end-to-end execution. *(3) Skill Construction:* The agent constructs reusable skills from upstream resources, such as documents or community materials. This setting evaluates the ability to transform raw resources into operational skills that improve subsequent task solving, rather than merely consuming pre-existing skills. *(4) Skill Composition:* The agent operates in a realistic wild setting with access to a large-scale skill repository, and must retrieve relevant skills and organize them into a valid execution workflow, including dependency ordering and cross-skill handoffs. This setting evaluates skill selection and orchestration in the wild. These four settings define a unified progression from no external skill use, to skill construction, to skill retrieval and orchestration, and finally to skill effectiveness. This formulation enables controlled comparison across different stages of skill use while remaining aligned with dynamically evolving real-world skill ecosystems.

3 SKILLNET-GYM CONSTRUCTION

3.1 DESIGN PRINCIPLES

To reflect the demands of a realistic and continuously evolving community, we design SkillNet-Gym around four key properties: *(1) Real-World Source:* SkillNet-Gym is grounded in artifacts collected from real community resources, rather than manually curated skills or synthetic files. This design choice ensures that the benchmark better reflects the structure, variability, and messiness of real-world usage scenarios. *(2) Domain diversity:* SkillNet-Gym spans 13 core domains and 81 subdomains, covering a broad range of practical settings, including data analysis, science, technology and so on, as illustrated in Figure 6. *(3) Dynamic extensibility:* To accommodate the continuously evolving community ecosystem, SkillNet is represented as a heterogeneous graph that can be incrementally expanded by adding new artifact nodes and their associated edges. This formulation enables the benchmark to evolve over time. *(4) Automatic construction:* Our pipeline automatically

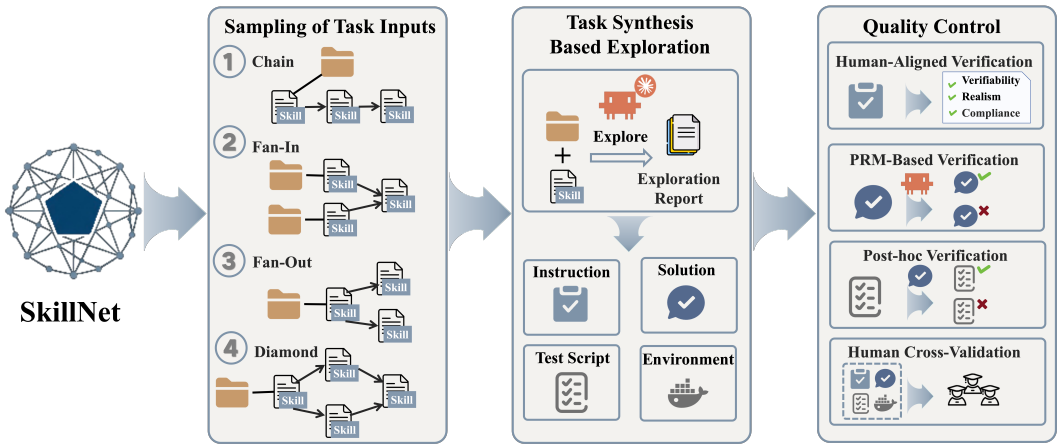


Figure 2: Overview of the SkillNet-Gym construction pipeline.

constructs the benchmark from SkillNet at scale, and also supports automatic control over task difficulty by varying the number of nodes visited during random walks and the topological structures induced by the traversal. The comparison of SkillNet-Gym with other benchmarks is detailed in Table 1. Next, we introduce the automated pipeline of SkillNet-Gym, covering SkillNet construction, benchmark data synthesis, and quality control. The pipeline is shown in Figure 2.

3.2 Construction of SkillNet

We construct SkillNet from real community resources by collecting three aligned artifact types: skills, supporting documents, and processable files. Specifically, we retrieve candidate skills from repositories such as ClawHub and SkillHub using LLM-generated domain-diverse queries, then perform filtering and deduplication. For each skill, we gather web documentation based on its functional semantics, or synthesize surrogate documentation when necessary, and retain functionally compatible files with source metadata for downstream normalization and quality control. On top of these artifacts, we build SkillNet as a skill-centric DAG: an LLM extracts each skill's pre and post scenarios, we identify candidate edges by matching embedded post-scenarios to pre-scenarios, and then use an LLM to verify whether each match constitutes a valid workflow handoff. The resulting graph captures directed compositional dependencies among skills and, compared with scenario-centric formulations, more directly models skill topology for multi-skill task synthesis.

3.3 Automatic Task Synthesis from SkillNet

Diversity Sampling of Task Inputs. Given the constructed SkillNet, we synthesize task candidates by sampling structured subgraphs from the directed skill graph. We consider five task patterns: single-skill tasks, chains, fan-out, fan-in, and diamonds, involving one to five skills depending on the structure, as illustrated in Figure 8. Each candidate preserves its participating skills, directed skill connections, and scenario-level evidence for every edge. We score each edge using alignment confidence, retrieval similarity, and the number of supporting scenario alignments, and score each task by combining mean edge quality with a structure bonus and a hub-node penalty. The resulting candidates are grouped by structure category and assigned difficulty labels from easy to expert according to their compositional complexity. We then select tasks by quality ranking. This produces a pool of compositional task candidates grounded in verified SkillNet transitions.

Agent-Driven Exploration Report Generation. Given the sampled skill Directed Acyclic Graph (DAG) and its associated input artifacts, we prompt an agent (Claude Code) to execute the skills by following the workflow specified by the DAG and operating directly on the input files. After the exploration phase, the agent produces an exploration report that summarizes actionable findings, such as feasible cross-skill workflows, potential intermediate artifacts, and common failure patterns. Compared with naively concatenating the full contents of all skill files, the exploration report is substantially more concise and better aligned with downstream task requirements.

Task Synthesis. A complete task should include at least four core components: a task instruction, a golden solution, an evaluation script, and the execution environment. We synthesize these components sequentially. *1) Task Instruction*: Given the exploration report and snippet-level information about the input artifacts, we prompt an LLM to generate a task instruction that follows the workflow induced by the skill DAG. To encourage diversity, each synthesis run uses a randomly sampled writing style. *2) Golden Solution*: After generating the instruction, we first derive high-level guidance from the exploration report to scaffold the downstream problem-solving process. We then prompt Claude Code to solve the task conditioned on this meta-information. From the resulting execution trajectory, we extract and compress the essential steps into a canonical reference solution, `solve.sh`. *3) Test Script*: Based on the task instruction and the final output artifacts, we synthesize a test script `test.sh` that verifies not only the structural validity of the outputs but also their semantic correctness at the value level. *4) Execution Environment*: Finally, we extract the minimal execution environment to get a `Dockerfile` from `solve.sh`.

3.4 Quality Control

We apply rigorous multi-stage quality control to the benchmark. First, during SkillNet construction, we filter low-quality nodes and edges to improve the reliability of the underlying skill graph. Second, for synthesized tasks, task-level quality control covers four core components of each instance: the instruction, the reference solution, the test script, and the task difficulty. Through this rigorous quality control pipeline, we ultimately retain 159 high-quality tasks. Detailed quality control procedures are provided in Appendix D.1.

3.5 Difficulty Stratification for Analysis

For analysis, we further partition the tasks into two coarse difficulty strata, **Easy** (77 tasks) and **Hard** (82 tasks), using post hoc empirical performance as a calibration signal while maintaining balanced domain coverage across the two subsets. This stratification enables more fine-grained comparisons of model behavior across tasks with different realized levels of difficulty.

4 Evaluation

4.1 Evaluation Methodology and Metrics

Skill Construction. We evaluate agents across three skill lifecycle settings: *(1) No Skill:* Agents solve tasks without skills. *(2) Official Skill:* Agents use community provided skills from repositories such as SkillHub and ClawHub. *(3) Skill-Creator:* Agents use a meta-skill (e.g., Skill-Creator) to distill human-authored procedural documents into reusable skills for task execution. All settings are evaluated end-to-end, with a binary reward indicating whether the final output passes all unit tests. We report **Avg@**k, the mean reward over k independent runs per task, averaged across tasks.

Skill Composition Beyond end-to-end evaluation, we assess skill retrieval and orchestration without requiring full task execution. Agents select relevant skills from a pool of 2,000+ SkillNet skills (Appendix D.2) and organize them into an execution workflow. We report two stage-specific metrics: (1) Skill Retrieval Completeness, indicating whether retrieved skills cover the entire gold skill set; and (2) Skill Orchestration Completeness, indicating whether predicted dependency edges fully recover the gold workflow among retrieved skills. Both metrics are binary, emphasizing recovery of a complete executable composition rather than partial overlap.

4.2 Experimental settings

Agent Harnesses. We evaluate the two most popular agents in current use: Claude Code (Anthropic, 2025) and Codex CLI (OpenAI, 2025a).

Models. We evaluate eight frontier models, including four open-source models and four proprietary models: MiniMax-M2.7 (MiniMax, 2026), Kimi-K2.6 (Team et al., 2026), Deepseek-V4 (DeepSeek-AI, 2026), GLM-5.1 (GLM-5-Team et al., 2026), Claude Haiku 4.5, Claude Sonnet 4.6, GPT-5.4-mini, and GPT-5.4. All models are evaluated with their default temperature settings, and

Models	*Skill Construction*						*Skill Composition*			
	No Skill		*Official Skill*		*Skill-Creator*		*Skills Retrieve*		*Skills Orchestration*	
	Easy	*Hard*	*Easy*	*Hard*	*Easy*	*Hard*	*Easy*	*Hard*	*Easy*	*Hard*
Claude Code										
Minimax-M2.7	61.04	17.89	65.58	26.22	66.88	20.73	8.12	13.10	18.55	11.62
Kimi-K2.6	70.13	26.83	74.03	34.15	73.38	31.10	14.29	19.51	4.62	12.66
Deepseek-V4	65.28	23.38	69.44	29.87	68.83	29.88	24.68	30.49	**22.22**	24.36
GLM-5.1	70.13	26.83	71.43	32.93	**79.22**	**31.10**	16.02	25.20	8.72	20.68
Claude 4.5 Haiku	59.09	16.46	55.84	26.22	53.90	23.17	15.21	21.65	8.72	9.33
Claude 4.6 Sonnet	62.34	20.73	74.03	36.59	72.73	30.49	**25.98**	**39.17**	14.59	**26.70**
Codex										
GPT-5.4-mini	72.73	26.83	74.03	32.93	75.32	30.49	5.63	14.63	6.67	15.19
GPT-5.4	**74.68**	**28.66**	**75.32**	**37.20**	74.03	26.83	11.26	18.29	11.35	16.46

Table 2: Comparison across different harnesses and models under different skill usage settings. **No Skill** uses no external skills; **Official Skill** directly provides the task-associated official skills; **Skill-Creator** evaluates execution with skills constructed from upstream resources; **Skills Retrieve** requires agents to retrieve relevant skills from a large wild skill pool before execution; **Skills Orchestration** requires agents to retrieve skills further organize into a dependency-aware workflow. Best and second-best results within each column are shown in **bold** and underlined, respectively.

each task is independently run three times. Following common practice, we pair open-source and Claude models with Claude Code, and GPT models with Codex.

4.3 Main Results

Skills Help Hard Tasks More Reliably Than Easy Tasks. Compared with the easy tasks, skill augmentation generally yields larger and more consistent relative improvements on the hard tasks. A likely reason is that current frontier models already possess strong agentic capabilities, even without external skills, they can often solve many easy tasks reasonably well. Consequently, the headroom for improvement on easy tasks is limited, whereas challenging tasks benefit more substantially from explicit skill support. The corresponding results are reported in Table 2.

One-Shot Skill Construction May Introduce Information Loss. Although the model is provided with documents of higher information density, compressing them into a skill inevitably discards part of the original information, which can lead to degraded performance for tasks requiring specific capabilities. The corresponding evidence is shown in Table 2. At the same time, our results also suggest that officially curated skills are not always superior to self-constructed ones, highlighting the importance of model-side adaptation. In addition, we evaluate another skill-construction baseline, SkillSeeker (Karaaslan, 2026), whose results are reported in Table 4 and exhibit similar trends. While our current study only considers a single-pass compression strategy, existing skill-construction methods are increasingly moving toward more sophisticated pipelines (Chen et al., 2026a; Zhang et al., 2026b; Si et al., 2026) that enable adaptation and self-evolution. We believe that, in the future, model constructed skills may eventually surpass currently human curated skills.

Frontier Agents Exhibit Weak Skill Composition Ability in the Wild. Although frontier agents can benefit from provided skills in controlled settings, their ability to operate over large, community-sourced skill repositories remains limited. Table 2 reveals a notable difficulty inversion between skill retrieval and skill orchestration. Hard tasks often obtain higher retrieval scores than easy tasks, suggesting that complex instructions expose richer procedural cues and make the required skills more identifiable. However, this advantage does not translate into successful orchestration. Once the agent must organize the retrieved skills into a dependency-preserving workflow, hard tasks become substantially more challenging. We also evaluate end-to-end execution in a realistic wild setting in Table 5. Compared with directly providing official skills, exposing agents to a large skill library leads to clear performance degradation, with the drop being more pronounced on hard tasks. These results show that incomplete retrieval and incorrect orchestration are not only intermediate failures, but directly limit downstream task success. Therefore, effective real-world skill use requires agents to go beyond consuming fixed skills and develop adaptive orchestration mechanisms that can expand exploration, verify dependencies, and update skill workflows as the skill ecosystem evolves.

Methods	Complete	Recall	Precision	Avg sel.
Skills Retrieve				
Agent	18.83	44.93	37.92	3.86
AgentSkillOS	42.14	61.77	**55.00**	-
SkillRouter				
w/o Reranker	36.48	60.28	19.06	-
w/ Reranker	44.03	67.40	21.01	-
SkillRet	**59.12**	**78.40**	24.09	-
Skills Orchestration				
Agent	18.36	25.17	18.63	3.27
AgentSkillOS	**29.49**	**35.51**	**26.86**	-

Table 3: Performance of Skill Composition Baselines.

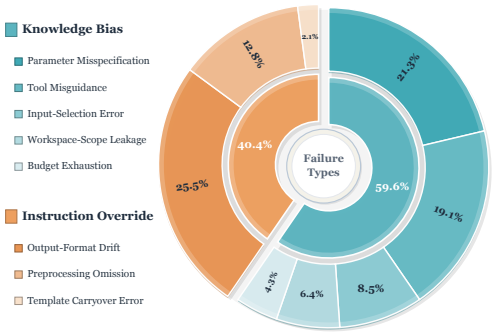


Figure 3: Analysis of Failure Types.

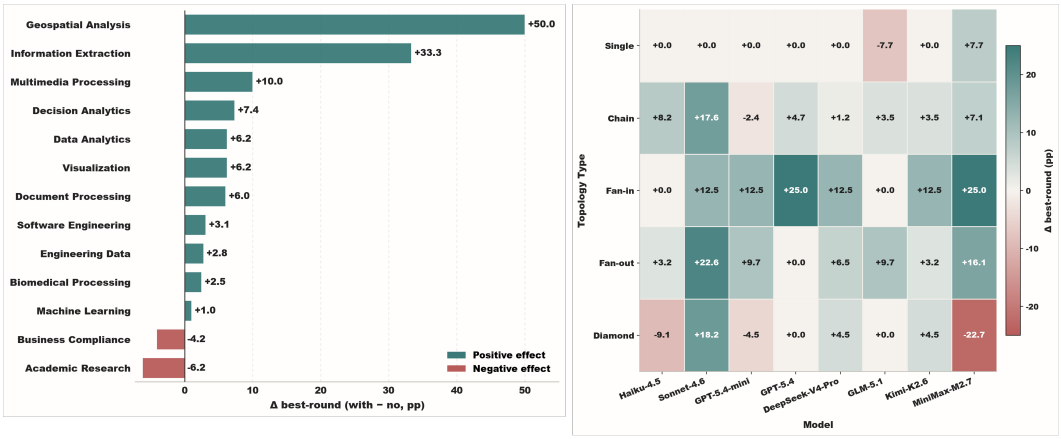
4.4 Analysis

Specialized Composition Baselines Remain Stronger than Agents. We further evaluate representative baselines specifically designed for skill composition, including SkillRouter (Zheng et al., 2026), SkillRet (Cho et al., 2026), and AgentSkillOS (Li et al., 2026a), and compare them against the average performance of the eight agents on Top-10. Average selection count denotes the average number of retrieved skills or selected dependency edges for the respective stage. SkillRet achieves substantially higher recall than SkillRouter despite the latter's reranker, possibly due to its larger training dataset. For AgentSkillOS, we replace its costly tree-construction pipeline with GLM-5.1 selection and orchestration over SkillRet candidates. This adapted approach substantially outperforms direct orchestration from the skill repository, highlighting the value of combining retrieval and orchestration for robust skill composition in the wild.

Analysis of Failure Types. We analyze bad cases, defined as tasks where skill injection strictly reduces agent accuracy, through manual inspection followed by AI-assisted analysis. Failures fall into two categories (Figure 3). *(1) Knowledge Bias (59.6%).* Injected skills provide incorrect or misleading guidance, including *tool misguidance* (inappropriate libraries, versions, or APIs), *parameter misspecification* (incorrect thresholds, regular expressions, or comparison directions), and *input selection errors* (wrong records or sub-tables). Other failures involve *workspace-scope leakage* from hard-coded paths and *budget exhaustion* from extra subroutines that truncate execution. These findings suggest that **skill construction should be adaptive to reduce the mismatch between human-curated skills and the model's actual execution behavior. Effective self-adaptation typically requires a dynamic evaluation benchmark to guide optimization, making SkillNet-Gym a useful testbed.** *(2) Instruction Override (40.4%).* Injected skills override explicit task requirements, causing *output-format drift* (incorrect JSON schemas, data types, or sign conventions), *preprocessing omission* (skipped normalization or editing steps), and *template carryover* (incomplete skill-provided scaffolding). These failures suggest that **current foundation models still struggle to balance contextual knowledge against direct instruction following.** Frontier models may need to internalize skill utilization as a native capability while preserving adherence to task instructions.

Skill Benefits Are Highly Domain Dependent. Skill utility varies across domains (Figure 4(a)). Geospatial Analysis (+50.0 pp) and Information Extraction (+33.3 pp) benefit most, suggesting that skills effectively capture reusable procedural workflows. Analytics and visualization show moderate gains, potentially through reduced tool-use overhead. Conversely, Academic Research (-6.2 pp) and Business Compliance (-4.2 pp) show slight declines, possibly because rigid procedures constrain open-ended exploration and adaptive planning. These results suggest that skill utility depends on domain-specific procedural regularity.

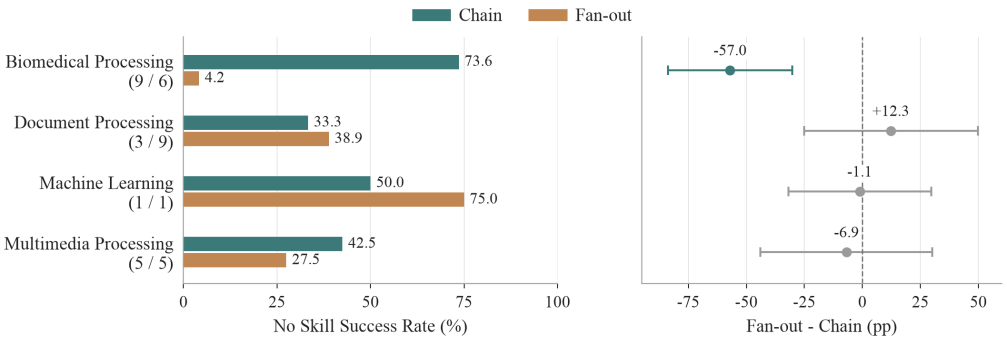
Task Topology Determines Skill Effectiveness. Task topology exhibits a clear correlation with skill effectiveness (Figure 4(b)). Single skill tasks receive almost no benefit, indicating that isolated procedures can already be handled reliably by modern foundation models. As dependency structures



(a) Skill Effect by Domain (b) Skill Effect by Topology Type

Figure 4: **Skill utility across domains and task topologies. (a) Domain-wise Skill Effect:** The bar chart reports the average performance delta of skill augmentation over the no skill setting. **(b) Topology-wise skill effect:** The heatmap shows the mean delta for each topology across eight models, revealing that skill effectiveness depends jointly on task structure and model capability.

become more compositional, performance improvements increase substantially. Chain, Fan-out, and especially Fan-in topologies consistently obtain the largest gains across most models, suggesting that skills are particularly effective at encoding reusable coordination patterns across multiple dependent subtasks. Diamond tasks, however, display the greatest variance, with strong models benefiting considerably while weaker models occasionally suffering performance degradation. This observation indicates that executing highly compositional skills requires not only well-structured procedural knowledge but also sufficient instruction-following and coordination capabilities from the underlying model. Overall, skills can provide their greatest value when tasks involve complex compositions of reusable operations rather than isolated actions.



(a) Performance by Domain (b) Adjusted Topology Contrast

Figure 5: **Task topology and empirical difficulty. (a) Domain-wise performance:** No skill success rates averaged across eight models. **(b) Adjusted topology contrast:** Fan-out vs chain gaps controlling for skill count and source batch, with clustered 95% confidence intervals.

Topology-Related Difficulty Is Context Dependent. The association between task topology and empirical difficulty varies across domains (Figure 5). Among 39 tasks involving three to five skills, fan-out tasks achieve much lower no-skill success than chains in Biomedical Processing (4.2% vs. 73.6%), whereas this pattern is absent in Document Processing (38.9% vs. 33.3%). These exploratory observations suggest that structural complexity does not define a universal difficulty ordering; its association with performance depends on the skill workflows involved. SkillNet-Gym should therefore calibrate task difficulty jointly using dependency structure, task semantics, and empirical agent performance.

Case Study: From SkillNet to Synthesized Tasks. We provide an example of the conversion from SkillNet to task instructions, as shown in Figure 9.

Furthermore, SkillNet-Gym can continuously synthesize new benchmarks and evolve together with the skill ecosystem.

5 Related Works

5.1 Benchmarking Agent Skills

Existing benchmarks on agent skills can be roughly grouped into three categories according to their evaluation focus. *(i) Skill Efficacy*: examine whether providing agents with external skills improves downstream task execution, typically by comparing no-skill and skill-augmented settings (Li et al., 2026b; Han et al., 2026; Ying et al., 2026). *(ii) Skill Construction*: benchmarks in this line focus on whether agents can construct or refine reusable skills from upstream resources or realistic task contexts (Zhong et al., 2026; Zhou et al., 2026b; Liu et al., 2026c; Chen et al., 2026b). *(iii) Skill Composition*: benchmarks in this line study how agents retrieve, select, route, and organize skills from large-scale skill pools to support task solving (Liu et al., 2026c; Li et al., 2026a; Cho et al., 2026; Zheng et al., 2026; Chen et al., 2026b).

Although these benchmarks provide valuable insights, they usually rely on fixed skill snapshots, or adopt separate protocols that make controlled comparison difficult. As a result, they offer limited support within dynamically evolving skill ecosystems. SkillNet-Gym addresses these limitations by dynamically constructing a community-grounded SkillNet and providing a dynamic benchmark for evaluating skill construction and skill composition.

5.2 Skill Usage in LLM Agents

Beyond benchmarking, a growing body of work studies how LLM agents acquire, retrieve, compose, and evolve skills during task solving. These studies can be roughly organized into four directions. *(i) Skill construction*: agents transform task trajectories, contextual information, or external resources into reusable skill units that can guide future problem solving (Ni et al., 2026; Wang et al., 2026a; Si et al., 2026; Liang et al., 2026; Xia et al., 2026a; Zhang et al., 2026c). *(ii) Skill retrieval*: agents retrieve relevant skills from a skill memory or skill bank and condition their policies on the retrieved knowledge to improve multi-step reasoning and task execution (Tu et al., 2026; Jiang et al., 2026; Zheng et al., 2026). *(iii) Skill orchestration*: agents organize retrieved skills into executable workflows or graph-structured plans, preserving dependencies among subtasks and enabling multi-skill execution (Fan et al., 2026; Xia et al., 2026b; Wang et al., 2026c; Chen et al., 2026a; Li et al., 2026c). *(iv) Skill evolution*: agents continuously refine, compose, and accumulate skills over repeated interactions, supporting long-term self-improvement (Ma et al., 2026; Zhang et al., 2026b; Alzubi et al., 2026; Ouyang et al., 2026; Wang et al., 2026b; Jiao et al., 2026; Yang et al., 2026; Liu et al., 2026a; Zhou et al., 2026a; Liu et al., 2026b; Ju et al., 2026; Zhang et al., 2026a). Different from these method-oriented studies, SkillNet-Gym focuses on constructing a dynamic benchmark that evaluates skill construction, skill composition, and skill efficacy under a unified protocol.

6 Conclusion

We introduce SkillNet-Gym, a unified and dynamic benchmark for evaluating skill use in LLM agents. Unlike most prior works that rely on manually constructed static benchmarks, SkillNet-Gym evolves alongside the skill ecosystem and continuously generates dynamic benchmarks. Our results show that skill utility depends on the evaluation setting: provided skills can help, particularly on hard tasks, while retrieval and orchestration remain important bottlenecks in large skill repositories. These findings suggest that effective skill use is not a single ability, but a pipeline of interdependent competencies. We hope SkillNet-Gym can serve as a realistic testbed for future research on skill-centric agents, dynamic benchmarks.

AI USE STATEMENT

In this work, we used generative AI tools as components of the SkillNet-Gym construction and evaluation pipeline. Specifically, LLMs were used to generate skill-retrieval queries, synthesize surrogate documentation when necessary, extract pre- and post-scenarios from skill specifications, and assess candidate inter-skill dependencies. We also used LLM-based agents to explore executable skill workflows and synthesize task instructions, reference solutions, evaluation scripts, and execution-environment specifications. Generative AI further supported quality control through rubric-based assessment of skills, task instructions, and test scripts, as well as trajectory-level verification and iterative refinement of candidate solutions. In our experiments, LLMs served as the evaluated agents for task solving, skill construction, and skill composition.

AI-generated benchmark artifacts were subjected to the multi-stage quality-control procedures described in Section 3 and Appendix D. These included execution-based checks of reference solutions against synthesized tests, assessment of test coverage and reliability, and preliminary model rollouts. The retained tasks were independently reviewed by three annotators with at least master's-level training in computer science, who assessed instruction clarity, reference-solution correctness, and test-script reliability; disagreements were resolved through discussion and majority vote. These checks mitigate, but do not eliminate, errors in AI-generated artifacts.

We take responsibility for the final content of this work, including all text, claims, code, and benchmark artifacts produced with the aid of generative AI.

REFERENCES

Salaheddin Alzubi, Noah Provenzano, Jaydon Bingham, Weiyuan Chen, and Tu Vu. Evoskill: Automated skill discovery for multi-agent systems, 2026. URL https://arxiv.org/abs/2603.02766.

Anthropic. Claude Code: an agentic coding tool. https://github.com/anthropics/claude-code, 2025.

Anthropic. Equipping agents for the real world with agent skills, October 2025. URL https://www.anthropic.com/engineering/equipping-agents-for-the-real-world-with-agent-skills. Anthropic Engineering Blog.

Chaithanya Bandi, Razvan-Gabriel Dumitru, Ben Hertzberg, Divyansh Agarwal, Geobio Boo, Tejas Polakam, Sami Hassaan, Jeff Da, HiJae Kim, Vipul Gupta, Manasi Sharma, Andrew Park, Martin Dimakis, Ernesto Gabriel Hernandez Montoya, Dan Rambado, Ivan Salazar, Rafael Cruz, MohammadHossein Rezaei, Chetan Rane, Ben Levin, Daniel Yue Zhang, Brad Kenstler, and Bing Liu. Mcp-atlas: A large-scale benchmark for tool-use competency with real mcp servers, 2026. URL https://arxiv.org/abs/2602.00933.

Le Chen, Erhu Feng, Yubin Xia, and Haibo Chen. Skvm: Revisiting language vm for skills across heterogenous llms and harnesses, 2026a. URL https://arxiv.org/abs/2604.03088.

Shiqi Chen, Jingze Gai, Ruochen Zhou, Jinghan Zhang, Tongyao Zhu, Junlong Li, Kangrui Wang, Zihan Wang, Zhengyu Chen, Klara Kaleb, Ning Miao, Siyang Gao, Cong Lu, Manling Li, Junxian He, and Yee Whye Teh. Skillcraft: Can llm agents learn to use tools skillfully?, 2026b. URL https://arxiv.org/abs/2603.00718.

Hongcheol Cho, Ryangkyung Kang, and Youngeun Kim. Skillret: A large-scale benchmark for skill retrieval in llm agents, 2026. URL https://arxiv.org/abs/2605.05726.

DeepSeek-AI. Deepseek-v4: Towards highly efficient million-token context intelligence, 2026.

Zhiyuan Fan, Tinghao Yu, Yuanjun Cai, Jiangtao Guan, Yun Yang, Dingxin Hu, Jiang Zhou, Xing Wu, Zhuo Han, Feng Zhang, and Lilin Wang. Toward scalable terminal task synthesis via skill graphs, 2026. URL https://arxiv.org/abs/2604.25727.

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.

Google. Gemini CLI: An open-source AI agent that brings the power of Gemini directly into your terminal. https://github.com/google-gemini/gemini-cli, 2025.

Tingxu Han, Yi Zhang, Wei Song, Chunrong Fang, Zhenyu Chen, Youcheng Sun, and Lijie Hu. Swe-skills-bench: Do agent skills actually help in real-world software engineering?, 2026. URL https://arxiv.org/abs/2603.15401.

Guanyu Jiang, Zhaochen Su, Xiaoye Qu, and Yi R. Fung. Xskill: Continual learning from experience and skills in multimodal agents, 2026. URL https://arxiv.org/abs/2603.12056.

Zhengbo Jiao, Shaobo Wang, Zifan Zhang, Xuan Ren, Wei Wang, Bing Zhao, Hu Wei, and Linfeng Zhang. Agentic proposing: Enhancing large language model reasoning via compositional skill synthesis, 2026. URL https://arxiv.org/abs/2602.03279.

Ruofei Ju, Xinrui Wang, Xin Ding, Yifan Yang, Hao Wu, Shiqi Jiang, Qianxi Zhang, Hao Wen, Xiangyu Li, Weijun Wang, Kun Li, Yunxin Liu, Haipeng Dai, Wei Wang, and Ting Cao. Embodiskill: Skill-aware reflection for self-evolving embodied agents, 2026. URL https://arxiv.org/abs/2605.10332.

Yusuf Karaaslan. Skill seekers, 2026. URL https://github.com/yusufkaraaslan/Skill_Seekers. Repository for converting documentation websites, GitHub repositories, and PDFs into Claude-compatible skills.

Kilo Code. Pinchbench: Real-world benchmarks for OpenClaw agents. https://github.com/pinchbench/skill, 2026. Benchmark skill repository, task definitions, and scoring scripts; leaderboard at https://pinchbench.com/.

Yoonho Lee, Roshen Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, and Chelsea Finn. Meta-harness: End-to-end optimization of model harnesses, 2026. URL https://arxiv.org/abs/2603.28052.

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, 2026a. URL https://arxiv.org/abs/2603.02176.

Junlong Li, Wenshuo Zhao, Jian Zhao, Weihao Zeng, Haoze Wu, Xiaochen Wang, Rui Ge, Yuxuan Cao, Yuzhen Huang, Wei Liu, Junteng Liu, Zhaochen Su, Yiyang Guo, Fan Zhou, Lueyang Zhang, Juan Michelini, Xingyao Wang, Xiang Yue, Shuyan Zhou, Graham Neubig, and Junxian He. The tool decathlon: Benchmarking language agents for diverse, realistic, and long-horizon task execution, 2025. URL https://arxiv.org/abs/2510.25726.

Xiangyi Li, Wenbo Chen, Yimin Liu, Shenghan Zheng, Xiaokun Chen, Yifeng He, Yubo Li, Bingran You, Haotian Shen, Jiankai Sun, Shuyi Wang, Binxu Li, Qunhong Zeng, Di Wang, Xuandong Zhao, Yuanli Wang, Roey Ben Chaim, Zonglin Di, Yipeng Gao, Junwei He, Yizhuo He, Liqiang Jing, Luyang Kong, Xin Lan, Jiachen Li, Songlin Li, Yijiang Li, Yueqian Lin, Xinyi Liu, Xuanqing Liu, Haoran Lyu, Ze Ma, Bowei Wang, Runhui Wang, Tianyu Wang, Wengao Ye, Yue Zhang, Hanwen Xing, Yiqi Xue, Steven Dillmann, and Han chung Lee. Skillsbench: Benchmarking how well agent skills work across diverse tasks, 2026b. URL https://arxiv.org/abs/2602.12670.

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, 2026c. URL https://arxiv.org/abs/2605.12039.

Qiliang Liang, Hansi Wang, Zhong Liang, and Yang Liu. From skill text to skill structure: The scheduling-structural-logical representation for agent skills, 2026. URL https://arxiv.org/abs/2604.24026.

Hongyi Liu, Haoyan Yang, Tao Jiang, Bo Tang, Feiyu Xiong, and Zhiyu Li. Skillsvote: Lifecycle governance of agent skills from collection, recommendation to evolution, 2026a. URL https://arxiv.org/abs/2605.18401.

Xingyan Liu, Xiyue Luo, Linyu Li, Ganghong Huang, Jianfeng Liu, and Honglin Qiao. Skillforge: Forging domain-specific, self-evolving agent skills in cloud technical support. *CoRR*, abs/2604.08618, 2026b. doi: 10.48550/ARXIV.2604.08618. URL https://doi.org/10.48550/arXiv.2604.08618.

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, 2026c. URL https://arxiv.org/abs/2604.04323.

Xinghua Lou, Miguel Lázaro-Gredilla, Antoine Dedieu, Carter Wendelken, Wolfgang Lehrach, and Kevin P. Murphy. Autoharness: improving llm agents by automatically synthesizing a code harness, 2026. URL https://arxiv.org/abs/2603.03329.

Ziyang Luo, Zhiqi Shen, Wenzhuo Yang, Zirui Zhao, Prathyusha Jwalapuram, Amrita Saha, Doyen Sahoo, Silvio Savarese, Caiming Xiong, and Junnan Li. Mcp-universe: Benchmarking large language models with real-world model context protocol servers, 2025. URL https://arxiv.org/abs/2508.14704.

Ziyu Ma, Shidong Yang, Yuxiang Ji, Xucong Wang, Yong Wang, Yiming Hu, Tongwen Huang, and Xiangxiang Chu. Skillclaw: Let skills evolve collectively with agentic evolver, 2026. URL https://arxiv.org/abs/2604.08377.

Mike A. Merrill, Alexander G. Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Yeon Shin, Thomas Walshe, E. Kelly Buchanan, Junhong Shen, Guanghao Ye, Haowei Lin, Jason Poulos, Maoyu Wang, Marianna Nezhurina, Jenia Jitsev, Di Lu, Orfeas Menis Mastromichalakis, Zhiwei Xu, Zizhao Chen, Yue Liu, Robert Zhang, Leon Liangyu Chen, Anurag Kashyap, Jan-Lucas Uslu, Jeffrey Li, Jianbo Wu, Minghao Yan, Song Bian, Vedang Sharma, Ke Sun, Steven Dillmann, Akshay Anand, Andrew Lanpouthakoun, Bardia Koopah, Changran Hu, Etash Guha, Gabriel H. S. Dreiman, Jiacheng Zhu, Karl Krauth, Li Zhong, Niklas Muennighoff, Robert Amanfu, Shangyin Tan, Shreyas Pimpalgaonkar, Tushar Aggarwal, Xiangning Lin, Xin Lan, Xuandong Zhao, Yiqing Liang, Yuanli Wang, Zilong Wang, Changzhi Zhou, David Heineman, Hange Liu, Harsh Trivedi, John Yang, Junhong Lin, Manish Shetty, Michael Yang, Nabil Omi, Negin Raoof, Shanda Li, Terry Yue Zhuo, Wuwei Lin, Yiwei Dai, Yuxin Wang, Wenhao Chai, Shang Zhou, Dariush Wahdany, Ziyu She, Jiaming Hu, Zhikang Dong, Yuxuan Zhu, Sasha Cui, Ahson Saiyed, Arinbjörn Kolbeinsson, Jesse Hu, Christopher Michael Rytting,

Ryan Marten, Yixin Wang, Alex Dimakis, Andy Konwinski, and Ludwig Schmidt. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces, 2026. URL https://arxiv.org/abs/2601.11868.

MiniMax. The minimax-m2 series: Mini activations unleashing max real-world intelligence. *CoRR*, abs/2605.26494, 2026. doi: 10.48550/ARXIV.2605.26494. URL https://doi.org/10.48550/arXiv.2605.26494.

Jingwei Ni, Yihao Liu, Xinpeng Liu, Yutao Sun, Mengyu Zhou, Pengyu Cheng, Dexin Wang, Erchao Zhao, Xiaoxi Jiang, and Guanjun Jiang. Trace2skill: Distill trajectory-local lessons into transferable agent skills, 2026. URL https://arxiv.org/abs/2603.25158.

OpenAI. Codex CLI: Lightweight coding agent that runs in your terminal. https://github.com/openai/codex, 2025a.

OpenAI. System Card for o3-mini, 2025b. URL https://openai.com/index/o3-mini-system-card/. Accessed on December 11, 2025.

Siru Ouyang, Jun Yan, Yanfei Chen, Rujun Han, Zifeng Wang, Bhavana Dalvi Mishra, Rui Meng, Chun-Liang Li, Yizhu Jiao, Kaiwen Zha, Maohao Shen, Vishy Tirumalashetty, George Lee, Jiawei Han, Tomas Pfister, and Chen-Yu Lee. Skillos: Learning skill curation for self-evolving agents, 2026. URL https://arxiv.org/abs/2605.06614.

Shuzheng Si, Haozhe Zhao, Yu Lei, Qingyi Wang, Dingwei Chen, Zhitong Wang, Zhenhailong Wang, Kangyang Luo, Zheng Wang, Gang Chen, Fanchao Qi, Minjia Zhang, and Maosong Sun. From context to skills: Can language models learn from context skillfully?, 2026. URL https://arxiv.org/abs/2604.27660.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, S. H. Cai, Yuan Cao, Y. Charles, H. S. Che, Cheng Chen, Guanduo Chen, Huarong Chen, Jia Chen, Jiahao Chen, Jianlong Chen, Jun Chen, Kefan Chen, Liang Chen, Ruijue Chen, Xinhao Chen, Yanru Chen, Yanxu Chen, Yicun Chen, Yimin Chen, Yingjiang Chen, Yuankun Chen, Yujie Chen, Yutian Chen, Zhirong Chen, Ziwei Chen, Dazhi Cheng, Minghan Chu, Jialei Cui, Jiaqi Deng, Muxi Diao, Hao Ding, Mengfan Dong, Mengnan Dong, Yuxin Dong, Yuhao Dong, Angang Du, Chenzhuang Du, Dikang Du, Lingxiao Du, Yulun Du, Yu Fan, Shengjun Fang, Qiulin Feng, Yichen Feng, Garimugai Fu, Kelin Fu, Hongcheng Gao, Tong Gao, Yuyao Ge, Shangyi Geng, Chengyang Gong, Xiaochen Gong, Zhuoma Gongque, Qizheng Gu, Xinran Gu, Yicheng Gu, Longyu Guan, Yuanying Guo, Xiaoru Hao, Weiran He, Wenyang He, Yunjia He, Chao Hong, Hao Hu, Jiaxi Hu, Yangyang Hu, Zhenxing Hu, Ke Huang, Ruiyuan Huang, Weixiao Huang, Zhiqi Huang, Tao Jiang, Zhejun Jiang, Xinyi Jin, Yu Jing, Guokun Lai, Aidi Li, C. Li, Cheng Li, Fang Li, Guanghe Li, Guanyu Li, Haitao Li, Haoyang Li, Jia Li, Jingwei Li, Junxiong Li, Lincan Li, Mo Li, Weihong Li, Wentao Li, Xinhang Li, Xinhao Li, Yang Li, Yanhao Li, Yiwei Li, Yuxiao Li, Zhaowei Li, Zheming Li, Weilong Liao, Jiawei Lin, Xiaohan Lin, Zhishan Lin, Zichao Lin, Cheng Liu, Chenyu Liu, Hongzhang Liu, Liang Liu, Shaowei Liu, Shudong Liu, Shuran Liu, Tianwei Liu, Tianyu Liu, Weizhou Liu, Xiangyan Liu, Yangyang Liu, Yanming Liu, Yibo Liu, Yuanxin Liu, Yue Liu, Zhengying Liu, Zhongnuo Liu, Enzhe Lu, Haoyu Lu, Zhiyuan Lu, Junyu Luo, Tongxu Luo, Yashuo Luo, Long Ma, Yingwei Ma, Shaoguang Mao, Yuan Mei, Xin Men, Fanqing Meng, Zhiyong Meng, Yibo Miao, Minqing Ni, Kun Ouyang, Siyuan Pan, Bo Pang, Yuchao Qian, Ruoyu Qin, Zeyu Qin, Jiezhong Qiu, Bowen Qu, Zeyu Shang, Youbo Shao, Tianxiao Shen, Zhennan Shen, Juanfeng Shi, Lidong Shi, Shengyuan Shi, Feifan Song, Pengwei Song, Tianhui Song, Xiaoxi Song, Hongjin Su, Jianlin Su, Zhaochen Su, Lin Sui, Jinsong Sun, Junyao Sun, Tongyu Sun, Flood Sung, Yunpeng Tai, Chuning Tang, Heyi Tang, Xiaojuan Tang, Zhengyang Tang, Jiawen Tao, Shiyuan Teng, Chaoran Tian, Pengfei Tian, Ao Wang, Bowen Wang, Chensi Wang, Chuang Wang, Congcong Wang, Dingkun Wang, Dinglu Wang, Dongliang Wang, Feng Wang, Hailong Wang, Haiming Wang, Hengzhi Wang, Huaqing Wang, Hui Wang, Jiahao Wang, Jinhong Wang, Jiuzheng Wang, Kaixin Wang, Linian Wang, Qibin Wang, Shengjie Wang, Shuyi Wang, Si Wang, Wei Wang, Xiaochen Wang, Xinyuan Wang, Yao Wang, Yejie Wang, Yipu Wang, Yiqin Wang, Yucheng Wang, Yuzhi Wang, Zhaoji Wang, Zhaowei Wang, Zhengtao Wang, Zhexu Wang, Zihan Wang, Zizhe Wang, Chu Wei, Ming Wei, Chuan Wen, Zichen Wen, Chengjie Wu, Haoning Wu, Junyan Wu, Rucong Wu, Wenhao Wu, Yuefeng Wu, Yuhao Wu, Yuxin Wu, Zijian Wu,

Chenjun Xiao, Jin Xie, Xiaotong Xie, Yuchong Xie, Yifei Xin, Bowei Xing, Boyu Xu, Jianfan Xu, Jing Xu, Jinjing Xu, L. H. Xu, Lin Xu, Suting Xu, Weixin Xu, Xinbo Xu, Xinran Xu, Yangchuan Xu, Yichang Xu, Yuemeng Xu, Zelai Xu, Ziyao Xu, Junjie Yan, Yuzi Yan, Guangyao Yang, Hao Yang, Junwei Yang, Kai Yang, Ningyuan Yang, Ruihan Yang, Xiaofei Yang, Xinlong Yang, Ying Yang, Yi Yang, Yi Yang, Zhen Yang, Zhilin Yang, Zonghan Yang, Haotian Yao, Dan Ye, Wenjie Ye, Zhuorui Ye, Bohong Yin, Chengzhen Yu, Longhui Yu, Tao Yu, Tianxiang Yu, Enming Yuan, Mengjie Yuan, Xiaokun Yuan, Yang Yue, Weihao Zeng, Dunyuan Zha, Haobing Zhan, Dehao Zhang, Hao Zhang, Jin Zhang, Puqi Zhang, Qiao Zhang, Rui Zhang, Xiaobin Zhang, Y. Zhang, Yadong Zhang, Yangkun Zhang, Yichi Zhang, Yizhi Zhang, Yongting Zhang, Yu Zhang, Yushun Zhang, Yutao Zhang, Yutong Zhang, Zheng Zhang, Chenguang Zhao, Feifan Zhao, Jinxiang Zhao, Shuai Zhao, Xiangyu Zhao, Yikai Zhao, Zijia Zhao, Huabin Zheng, Ruihan Zheng, Shaojie Zheng, Tengyang Zheng, Junfeng Zhong, Longguang Zhong, Weiming Zhong, M. Zhou, Runjie Zhou, Xinyu Zhou, Zaida Zhou, Jinguo Zhu, Liya Zhu, Xinhao Zhu, Yuxuan Zhu, Zhen Zhu, Jingze Zhuang, Weiyu Zhuang, Ying Zou, and Xinxing Zu. Kimi k2.5: Visual agentic intelligence, 2026. URL https://arxiv.org/abs/2602.02276.

Songjun Tu, Chengdong Xu, Qichao Zhang, Yaocheng Zhang, Xiangyuan Lan, Linjing Li, and Dongbin Zhao. Dynamic dual-granularity skill bank for agentic rl, 2026. URL https://arxiv.org/abs/2603.28716.

Chenxi Wang, Zhuoyun Yu, Xin Xie, Wuguannan Yao, Runnan Fang, Shuofei Qiao, Kexin Cao, Guozhou Zheng, Xiang Qi, Peng Zhang, and Shumin Deng. Skillx: Automatically constructing skill knowledge bases for agents, 2026a. URL https://arxiv.org/abs/2604.04804.

Jianing Wang, Linsen Guo, Zhengyu Chen, Qi Guo, Hongyu Zang, Wenjie Shi, Haoxiang Ma, Xiangyu Xi, Xiaoyu Li, Wei Wang, and Xunliang Cai. Heavyskill: Heavy thinking as the inner skill in agentic harness, 2026b. URL https://arxiv.org/abs/2605.02396.

Jiayu Wang, Yifei Ming, Zixuan Ke, Shafiq Joty, Aws Albarghouthi, and Frederic Sala. Skillorchestra: Learning to route agents via skill transfer, 2026c. URL https://arxiv.org/abs/2602.19672.

Peng Xia, Jianwen Chen, Hanyang Wang, Jiaqi Liu, Kaide Zeng, Yu Wang, Siwei Han, Yiyang Zhou, Xujiang Zhao, Haifeng Chen, Zeyu Zheng, Cihang Xie, and Huaxiu Yao. Skillrl: Evolving agents via recursive skill-augmented reinforcement learning, 2026a. URL https://arxiv.org/abs/2602.08234.

Tianle Xia, Lingxiang Hu, Yiding Sun, Ming Xu, Lan Xu, Siying Wang, Wei Xu, and Jie Jiang. Grasp: Graph-structured skill compositions for llm agents, 2026b. URL https://arxiv.org/abs/2604.17870.

Yutao Yang, Junsong Li, Qianjun Pan, Bihao Zhan, Yuxuan Cai, Lin Du, Jie Zhou, Kai Chen, Qin Chen, Xin Li, Bo Zhang, and Liang He. Autoskill: Experience-driven lifelong learning via skill self-evolution, 2026. URL https://arxiv.org/abs/2603.01145.

Bowen Ye, Rang Li, Qibin Yang, Yuanxin Liu, Linli Yao, Hanglong Lv, Zhihui Xie, Chenxin An, Lei Li, Lingpeng Kong, Qi Liu, Zhifang Sui, and Tong Yang. Claw-eval: Towards trustworthy evaluation of autonomous agents. arXiv preprint arXiv:2604.06132, 2026. URL https://arxiv.org/abs/2604.06132. Leaderboard and code: https://claw-eval.github.io/.

Jiahao Ying, Boxian Ai, Wei Tang, Siyuan Liu, and Yixin Cao. Openskilleval: Automatically auditing the open skill ecosystem for llm agents, 2026. URL https://arxiv.org/abs/2605.23657.

Genrui Zhang, Erle Zhu, Jinfeng Zhou, Caiyan Jia, and Hongning Wang. Skillevolver: Skill learning as a meta-skill, 2026a. URL https://arxiv.org/abs/2605.10500.

Hanrong Zhang, Shicheng Fan, Henry Peng Zou, Yankai Chen, Zhenting Wang, Jiayu Zhou, Chengze Li, Wei-Chieh Huang, Yifei Yao, Kening Zheng, Xue Liu, Xiaoxiao Li, and Philip S. Yu. Coevoskills: Self-evolving agent skills via co-evolutionary verification, 2026b. URL https://arxiv.org/abs/2604.01687.

Mingda Zhang, Tiesunlong Shen, Haoran Luo, Wenjin Liu, Zikai Xiao, Erik Cambria, and Xiaoying Tang. Skillflow: Flow-driven recursive skill evolution for agentic orchestration, 2026c. URL https://arxiv.org/abs/2605.14089.

Yuxuan Zhang, Yubo Wang, Yipeng Zhu, Penghui Du, Junwen Miao, Xuan Lu, Wendong Xu, Yunzhuo Hao, Songcheng Cai, Xiaochen Wang, Huaisong Zhang, Xian Wu, Yi Lu, Minyi Lei, Kai Zou, Huifeng Yin, Ping Nie, Liang Chen, Dongfu Jiang, Wenhu Chen, and Kelsey R. Allen. Clawbench: Can ai agents complete everyday online tasks?, 2026d. URL https://arxiv.org/abs/2604.08523.

YanZhao Zheng, ZhenTao Zhang, Chao Ma, YuanQiang Yu, JiHuai Zhu, Yong Wu, Tianze Xu, Baohua Dong, Hangcheng Zhu, Ruohui Huang, and Gang Yu. Skillrouter: Skill routing for llm agents at scale, 2026. URL https://arxiv.org/abs/2603.22455.

Shanshan Zhong, Yi Lu, Jingjie Ning, Yibing Wan, Lihan Feng, Yuyi Ao, Leonardo F. R. Ribeiro, Markus Dreyer, Sean Ammirati, and Chenyan Xiong. Skilllearnbench: Benchmarking continual learning methods for agent skill generation on real-world tasks, 2026. URL https://arxiv.org/abs/2604.20087.

Huichi Zhou, Siyuan Guo, Anjie Liu, Zhongwei Yu, Ziqin Gong, Bowen Zhao, Zhixun Chen, Menglong Zhang, Yihang Chen, Jinsong Li, Runyu Yang, Qiangbin Liu, Xinlei Yu, Jianmin Zhou, Na Wang, Chunyang Sun, and Jun Wang. Memento-skills: Let agents design agents, 2026a. URL https://arxiv.org/abs/2603.18743.

Yifan Zhou, Zhentao Zhang, Ziming Cheng, Shuo Zhang, Qizhen Lan, Zhangquan Chen, Zhi Yang, QianyuXu, Ronghao Chen, Huacan Wang, and Sen Hu. Skillgenbench: Benchmarking skill generation pipelines for llm agents, 2026b. URL https://arxiv.org/abs/2605.18693.

A Limitations

Dependence on Benchmark Construction. Although SkillNet-Gym is designed to be dynamically extensible, the quality of the benchmark still depends critically on the reliability of the automated construction pipeline. Skill collection, scenario extraction, edge verification, task synthesis, and test generation all involve LLM-based components. While we employ multi-stage quality control and human cross-validation, expert oversight remains necessary in highly specialized vertical domains, where subtle errors may be difficult to detect automatically.

Task limitations. SkillNet-Gym currently focuses on tasks whose outcomes can be verified deterministically through unit tests, and does not yet cover more open-ended task formats. This limitation suggests an important direction for future work: developing more flexible and robust testing scripts and evaluation protocols to better capture the broader range of real-world skill usage.

B Comparison Traits Details

We compare representative agentic skill benchmarks along eight traits spanning three dimensions: benchmark construction, benchmark scope, and evaluation focus.

- **Real-World Source**: Whether the benchmark is grounded in artifacts, repositories, or tasks drawn from real-world or community environments, rather than being primarily handcrafted, synthetic, or narrowly curated for evaluation convenience.
- **Automatic Construction**: Whether the benchmark can be constructed largely through an automated pipeline, such as automatic data collection, graph construction, task synthesis, or verifier generation, instead of relying mainly on manual authoring and curation.
- **Dynamic Extensibility**: Whether the benchmark can be incrementally updated as new skills, artifacts, or workflows emerge, allowing it to evolve with the underlying ecosystem rather than remaining a static snapshot.

- **Domain Coverage**: Whether the benchmark covers a sufficiently broad range of domains and task settings, enabling evaluation beyond a narrow vertical such as software engineering or a small fixed taxonomy.
- **Skill Pool**: Whether the benchmark includes a comparatively large candidate pool of skills or skill-like artifacts, such that agents must operate under realistic retrieval or selection conditions rather than relying only on pre-attached or task-specific skills.
- **Task Numbers**: The number of benchmark task instances available for evaluation. Larger task sets typically improve statistical reliability, diversity, and robustness of conclusions, although scale alone does not imply realism or breadth.
- **Skill Construction**: Whether the benchmark explicitly evaluates an agent's ability to create, generate, or refine new skills, rather than only consuming pre-existing skills during task solving.
- **Skill Composition**: Whether the benchmark explicitly evaluates how agents retrieve, organize, or compose multiple skills or tool-like units into coherent workflows, rather than treating skills as isolated single-use aids.

C Further Experiment Details

C.1 Construction Baselines

We additionally evaluate SkillSeeker (Karaaslan, 2026) under skill construction setting. The results are reported in Table 4. The results suggest that different skill construction methods can play complementary roles across models. Meanwhile, both methods largely rely on directly compressing documentation into reusable skills, their effectiveness is inherently limited. This observation points to a promising direction for future work: skill construction methods that go beyond static distillation and instead support iterative self-improvement and empirical validation.

Models	*Skill Creator*		*Skill Seeker*	
	Easy	*Hard*	*Easy*	*Hard*
Claude Code				
Minimax-M2.7	66.88	20.73	60.39	23.17
Kimi-K2.6	73.38	31.10	75.32	38.41
Deepseek-V4	68.83	29.88	72.73	23.17
GLM-5.1	79.22	31.10	70.78	28.05

Table 4: Performance of skill construction baselines.

C.2 End-to-End Performance in the Wild

We evaluate agents in a wild setting, where they are exposed to a large skill library containing over 2000 skills, and measure their task performance. The results are shown in Table 5. Compared with the setting where the agent is provided with the correct official skills, performance drops on both Easy and Hard tasks to varying degrees, with the degradation being particularly pronounced on Hard tasks. We attribute this decline to the difficulty of operating over a massive skill library: the agent often fails to retrieve relevant skills, or retrieves only an incomplete subset of them. These results further highlight that a strong skill composition baseline is crucial for enabling effective skill usage in realistic, open-world scenarios.

D Benchmark Details

D.1 Benchmark Construction Details

SkillNet Construction Details. We implement SkillNet construction as a staged pipeline that turns broad community skill retrieval results into a scenario-mediated directed skill graph. The

Models	*Official Skill*		*In Wild*	
	Easy	*Hard*	*Easy*	*Hard*
Claude Code				
Minimax-M2.7	65.58	26.22	65.58	23.17
Kimi-K2.6	74.03	34.15	66.23	25.61
Deepseek-V4	69.44	29.87	68.83	24.39
GLM-5.1	71.43	32.93	66.23	24.39

Table 5: End-to-End Task Performance of Agents in the Wild Setting.

pipeline first collects and materializes candidate skills, removes near-duplicates, abstracts each skill into pre- and post-scenarios, aligns compatible scenarios across skills, and finally aggregates verified alignments into `compose_with` edges for task synthesis.

Skill Retrieval and Materialization. SkillNet construction starts from broad query seeds covering diverse domains. For each query, we call the SkillNet search API in vector mode with a fixed similarity threshold and bounded top-k limit. Retrieved skills are ranked by GitHub stars and truncated to the top candidates, with optional deduplication by skill URL or name. The selected `skill_urls` are then resolved and downloaded from GitHub, producing a local skill corpus with a manifest that records provenance and download status.

Near-Duplicate Skill Removal. To reduce redundancy, each retained skill is embedded using its name, metadata, matched queries, and full `SKILL.md` content. A FAISS nearest-neighbor index identifies highly similar skills, which are merged with union-find clustering. The representative of each cluster is selected by GitHub stars with deterministic tie-breaking.

Scenario Extraction and Canonicalization. For each deduplicated skill, an LLM extracts scenario-mediated transitions from the skill specification. A pre-scenario describes the state before the skill is applicable, while a post-scenario describes the state after successful execution. Raw scenarios are normalized, embedded, and organized into a sparse FAISS similarity graph. Louvain community detection provides coarse groups, which are further refined by complete-linkage clustering. Each final cluster is assigned a canonical scenario name and linked back to its skills.

Scenario Alignment and Edge Verification. Cross-skill edge candidates are induced by retrieving compatible pre-scenarios for each post-scenario. Post-scenario embeddings query a FAISS index over pre-scenario embeddings, and candidates below the retrieval threshold are discarded. An LLM judge then verifies whether the source post-scenario can naturally satisfy the target pre-scenario in a real workflow.

Graph Assembly. The graph builder aggregates verified post-to-pre alignments into directed `compose_with` edges between skill nodes. Multiple alignments between the same source–target skill pair are grouped into one edge with accumulated evidence. Each edge stores the source and target skills, supporting scenarios, alignment types, confidence scores, retrieval scores, and scenario-level evidence. The final directed SkillNet graph is used for subgraph sampling and downstream task synthesis.

Quality Control for SkillNet. To ensure that SkillNet remains reliable and non-redundant, we apply quality control throughout its construction, covering skills, edges and the downstream artifacts. *1) Skill Level:* Each downloaded `SKILL.md` is evaluated by an LLM judge along three dimensions: environment cost, output verifiability, and documentation quality. A skill is retained only if it meets the minimum threshold for each dimension and also satisfies the overall weighted score requirement. We further remove near-duplicate skills by embedding the full skill descriptions, retrieving high-similarity neighbors with FAISS, clustering duplicates, and keeping the highest-star representative within each cluster. *2) Edge Level:* Candidate skill pairs are reviewed using both their full `SKILL.md` files and scenario-level evidence. We remove edges when the target skill merely restates the source capability, represents an alternative implementation of the same function, or fails to introduce a substantively new state transition. *3) Artifact Level:* Auxiliary documents and files are retained only if they are substantively relevant to the associated skill, which helps filter out loosely

related or noisy context. Together, these steps keep SkillNet compact, well-grounded, and effective for downstream task synthesis.

Quality Control for Synthesized Tasks. Task-level quality control covers four core components of each synthesized task: the instruction, the reference solution, the test script, and the task difficulty.

Human-Aligned Rubric Judge for Task Instructions. We design a rubric-based judge to assess the quality of synthesized task instructions along three key dimensions: *1) Verifiability*, whether the instruction admits a uniquely testable outcome under the evaluation script; *2) Human-expert Realism*, whether the instruction resembles one that a human expert would naturally write, rather than exhibiting an overly synthetic or machine-generated style; and *3) DAG compliance*, whether the instruction is consistent with the workflow specified by the sampled DAG. To further align the judge with authentic human preferences, we refine the prompting strategy of the LLM-as-a-Judge using manually curated benchmarks, such as SkillsBench, as supervision signals.

PRM-Based Verification of Ground-Truth Solutions. Although we obtain candidate solutions by prompting an agent with meta guidance, the resulting outputs are often difficult to verify directly and may otherwise require costly review by human experts. To address this issue, we employ an additional agent to monitor and assess the intermediate execution trajectory used to produce the solution. If this verifier agent judges the reasoning and execution process to be sound, we treat the resulting artifact as a trustworthy ground-truth solution. Importantly, we do not immediately discard samples whose trajectories are flagged as problematic, as doing so would incur substantial synthesis cost. Instead, we feed the failure signals and diagnostic feedback from unsuccessful attempts into the context of subsequent retries. This iterative refinement process continues until a valid solution is produced or a maximum retry budget is reached, at which point the sample is abandoned.

Test Script Generation with Post-hoc Verification. The synthesized evaluation script should satisfy two key requirements: it must be fully passed by the reference solution, and it must comprehensively cover all evaluation conditions implied by the task instruction. We iteratively refine the test script until these criteria are satisfied. In addition, to prevent the tests from being either too weak or overly restrictive, we design a rubric that allows an LLM to evaluate the quality of the test script itself. We also conduct preliminary rollouts with candidate models to assess whether the tests are under-constraining and fail to distinguish correct solutions from incomplete or incorrect ones.

Difficulty-Aware Filtering and Human Cross-Validation. To ensure that the synthesized tasks genuinely challenge an agent's ability to orchestrate complex skill workflows, we adopt a rigorous two-stage task retention pipeline. First, we apply difficulty-aware filtering to remove tasks that can be successfully solved by GLM-5.1 (GLM-5-Team et al., 2026) in three independent trials without skills. This step removes shortcut-prone tasks that can be completed from general prior knowledge alone. The remaining tasks are then independently reviewed by three annotators with at least master's-level training in computer science. Each annotator evaluates instruction clarity, ground-truth correctness, and test-script reliability. Disagreements are resolved by majority vote after discussion. Inter-annotator agreement is high, with Fleiss's Kappa $\kappa = 0.834$, indicating strong consistency within the range $0.80 \leq \kappa \leq 1.00$.

D.2 SKILLNET STATISTICS

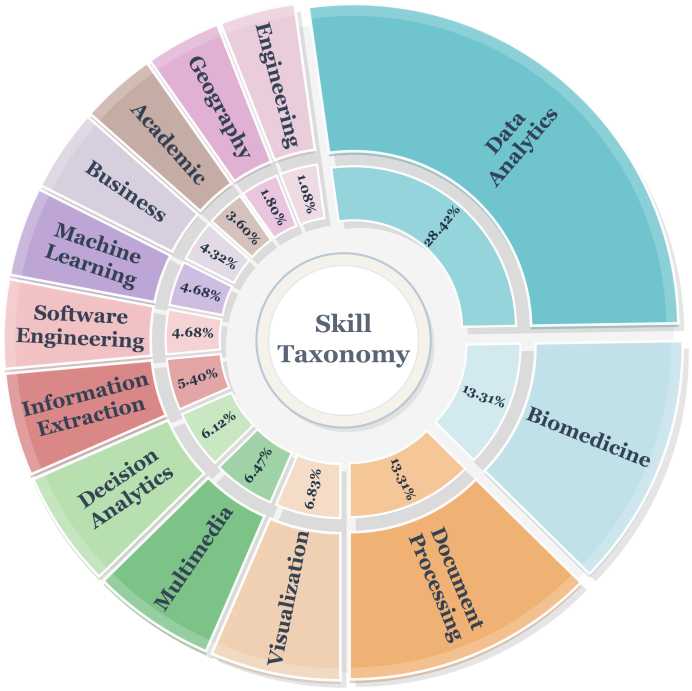


Figure 6: Distribution of primary skill categories in SkillNet.

Distribution of Skill Types in SkillNet. Figure 6 shows the proportion of skills in each primary category. Data Analytics accounts for the largest share (28.42%), followed by Biomedicine and Document Processing (13.31% each). The remaining categories span diverse domains, including Visualization, Multimedia, and Decision Analytics, highlighting the breadth of SkillNet.

SkillNet Construction Statistics. Table 6 reports the scale of SkillNet at different stages of construction. Starting from 3,142 query seeds, the pipeline downloads 8,159 candidate skills for quality evaluation. After quality filtering and deduplication, 2,225 skills are retained in the final SkillNet. The graph construction stage reduces 255,965 candidate edges to 53,904 final edges, with an average out-degree of 24.23. In addition, 285 entity files are downloaded as supporting artifacts for downstream task synthesis.

Statistic	Value
Query seeds	3,142
Skills entering quality evaluation after download	8,159
Skills retained after quality filtering	3,340
Final skills after deduplication	2,225
Candidate edges before filtering	255,965
Final graph edges	53,904
Average out-degree	24.23
Downloaded entity files	285

Table 6: Statistics of skill collection and graph construction.

D.3 Benchmark Task Statistics

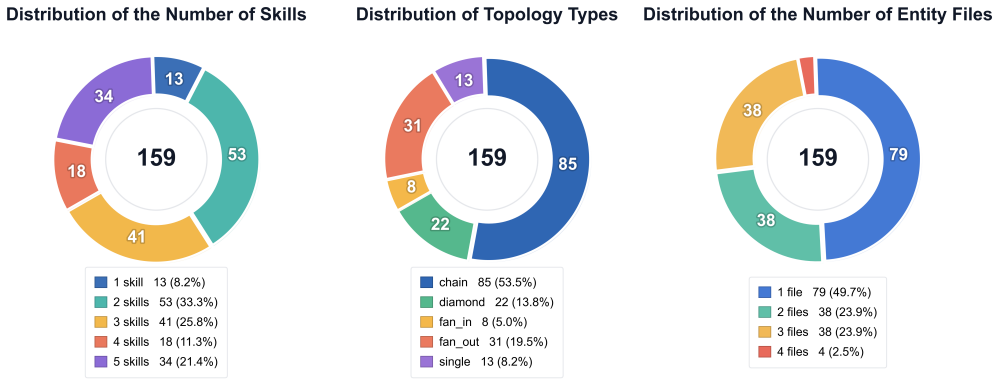


Figure 7: Statistics of synthesized benchmark tasks. The figure summarizes the synthesized benchmark tasks from three structural perspectives: skill count, workflow topology, and entity-file count. These statistics characterize how tasks are composed from SkillNet subgraphs and how many external artifacts are involved during task execution.

D.4 Task Topology

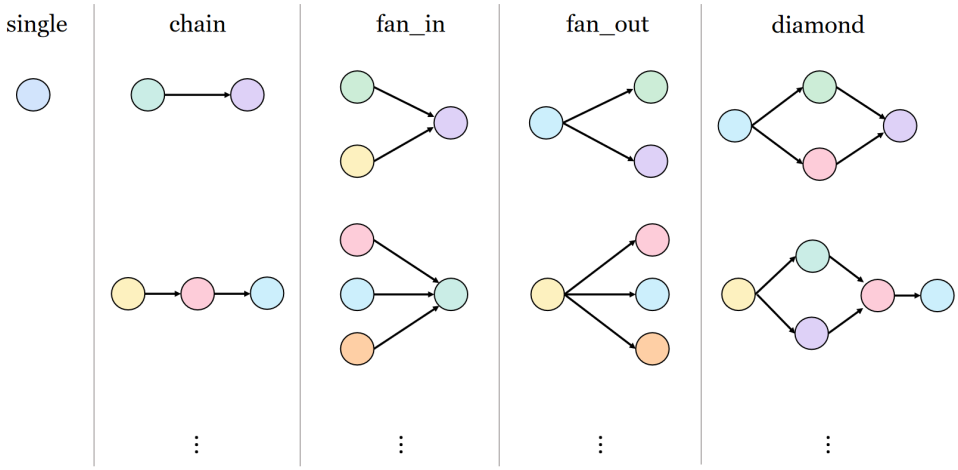


Figure 8: Illustration of task topology patterns used for synthesized tasks. The figure illustrates the task topology patterns used in our benchmark construction. A single task contains one isolated skill; a chain connects skills sequentially; fan-in merges multiple upstream skills into a downstream skill; fan-out branches one skill into multiple downstream skills; and diamond structures combine branching and merging patterns.

D.5 Skill Taxonomy

E Experiments Details

E.1 Baseline Details

SkillSeeker (Karaaslan, 2026). This toolkit constructs AI skills from real-world resources such as documentation, repositories, PDFs, and local codebases. It transforms these resources into structured knowledge assets for LLM agents and retrieval systems. The generated skills can provide domain-specific knowledge for downstream AI applications.

Category	Subcategories
Document Processing	PDF Extraction & OCR; Word/DOCX Processing; Excel/Spreadsheet Processing; Markdown/HTML/LaTeX Conversion; Document Summarization & QA; Document Comparison & Review; Document Metadata & Accessibility
Data Analytics	CSV/Table Cleaning; Data Validation & Quality Checks; ETL & Format Conversion; JSON/XML/YAML Processing; Statistical Analysis; Time Series Analysis; Survey, Census & Public Data Analysis
Visualization	Chart Generation; Dashboard & Interactive Visualization; Geospatial Visualization; Scientific Figure Generation; Business Report Generation; Automated Narrative Reporting; Accessibility-Oriented Visualization
Software Engineering	Code Review & Refactoring; Code Generation & Transformation; Testing & Test Case Generation; API Design, Testing & Documentation; Log Analysis & Debugging; Configuration & Infrastructure as Code; Software Documentation
Information Extraction	API Data Retrieval; HTML Parsing; Knowledge Extraction; Entity & Relation Extraction; Structured Data Extraction
Machine Learning	Tabular Machine Learning; Model Evaluation & Benchmarking; Feature Engineering; Classification & Regression; Clustering & Similarity Search; NLP & Text Classification; Data Leakage & ML Validation
Multimedia	Audio Transcription & Speech Processing; Video Processing & Summarization; Image Processing; Image Classification & Annotation; OCR for Images; Media Metadata Extraction; Media Accessibility
Geography	GIS File Processing; GeoJSON/Shapefile Processing; Hydrology & Water Quality Analysis; Environmental Monitoring; Spatial Statistics & Interpolation
Biomedicine	FASTA/FASTQ Sequence Processing; Variant Calling & Annotation; Genomics Data Analysis; Single-Cell & Omics Analysis; Protein & Structure Analysis; Molecular/SMILES Analysis; Medical Image Processing; Privacy Protection & De-identification
Decision Analytics	Mathematical Optimization; Scheduling & Resource Allocation; Planning & Constraint Solving; Simulation Result Analysis; Forecasting & Scenario Analysis; Numerical Methods & Stability Analysis; Power System Simulation
Engineering	CAD File Processing; Sensor & IoT Data Processing; Equipment Maintenance Analysis
Business	Customer Feedback & Support Analysis; Business Intelligence; Risk & Audit Analysis; Security Log & Vulnerability Analysis; Privacy, GDPR & Compliance; Regulatory Document Analysis; Legal & Contract Review
Academic	Research Paper Analysis; Educational Content Generation; Knowledge Graph Construction; Study Material Summarization

Table 7: Taxonomy of skill categories and subcategories of SkillNet-Gym.

SkillRouter (Zheng et al., 2026). This method studies skill routing in large-scale skill registries. Given a user task and a large pool of candidate skills, it retrieves and reranks the most relevant skills before downstream agent execution. It highlights the importance of using full skill text, rather than only names and descriptions, when selecting among highly overlapping skills.

SkillRet (Cho et al., 2026). This benchmark focuses on skill retrieval in large-scale LLM agent systems. It contains a curated library of public agent skills and natural-language queries, with structured taxonomy and separate training and evaluation splits. By isolating retrieval from downstream execution, it evaluates how well models can select relevant skills from realistic skill libraries.

AgentSkillOS (Li et al., 2026a). This framework addresses the organization and orchestration of agent skills at ecosystem scale. It organizes large skill collections into a capability tree for skill discovery, and composes selected skills through DAG-based execution plans. It also introduces an artifact-rich evaluation setting to assess how agents select, combine, and use skills for complex tasks.

F CASE STUDY

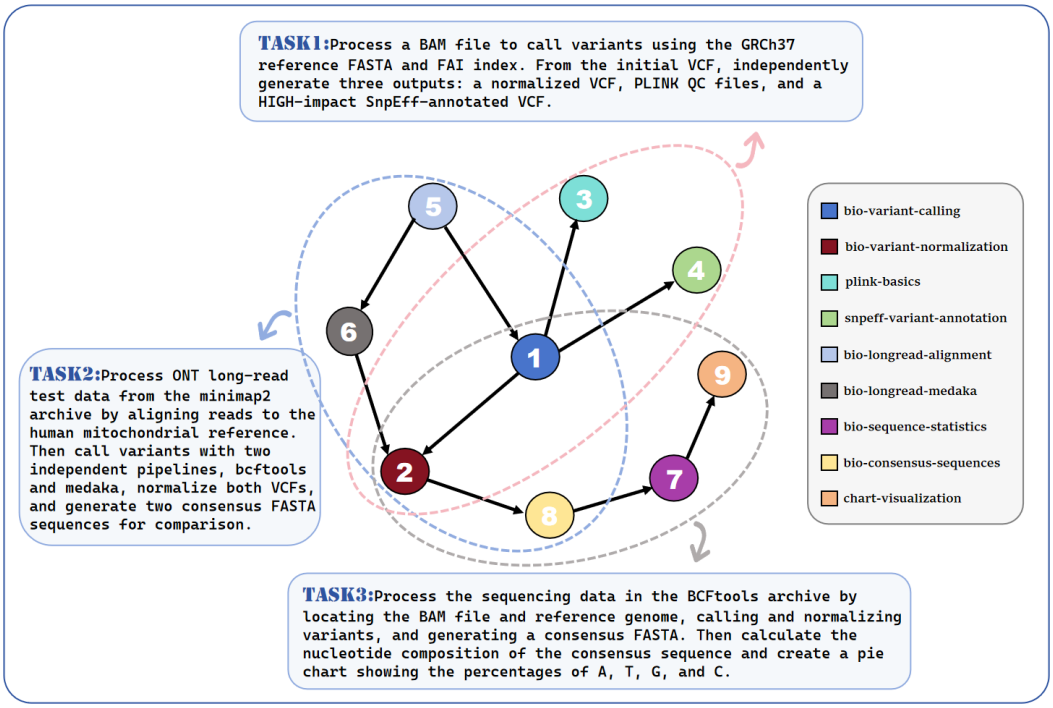


Figure 9: Case Study: From SkillNet subgraphs to synthesized tasks.

Skill	Description
`bio-variant-calling`	Calls SNPs and indels from aligned sequencing reads. It typically takes a BAM file and a reference FASTA as input, uses `bcftools mpileup` and `bcftools call`, and outputs VCF files.
`bio-variant-normalization`	Normalizes variant representations in VCF files, including left-aligning indels, splitting multiallelic sites, fixing REF allele mismatches, and removing duplicate records.
`plink-basics`	Performs PLINK-based population genetics processing, including format conversion among VCF, BED/BIM/FAM, PED/MAP, and PGEN files, as well as common QC filtering.
`snpeff-variant-annotation`	Annotates and filters variants in VCF files using SnpEff and SnpSift. It predicts variant effects, gene names, transcripts, amino acid changes, and HGVS representations.
`bio-longread-alignment`	Aligns Oxford Nanopore or PacBio long reads to a reference genome using `minimap2`, and outputs SAM, BAM, or PAF alignment files.
`bio-longread-medaka`	Performs Oxford Nanopore assembly polishing and variant calling with Medaka. It can generate polished consensus FASTA files or call variants from Nanopore reads.
`bio-sequence-statistics`	Computes statistics and QC summaries for FASTA or sequence datasets, including sequence count, total length, N50/L50, GC content, length distribution, and nucleotide composition.
`bio-consensus-sequences`	Applies VCF variants to a reference FASTA to generate sample-specific consensus sequences using `bcftools consensus`. It supports sample selection, masking, and region extraction.
`chart-visualization`	Converts structured data into visual charts such as line, bar, scatter, pie, heatmap, network, or flow diagrams, and exports visual results.

Table 8: Descriptions of skills used in the case study.

G PROMPT TEMPLATE

Skill Suitability Evaluation Prompt

You evaluate whether a downloaded skill is suitable for large-scale benchmark task construction.

Output Requirement
Return exactly one JSON object. Do not include markdown. Score each dimension from 1 to 5. Scores must be integers.

Evaluation Dimensions

1. `environment_cost_score`
This score measures how lightweight and reproducible the skill is. Higher is better.
- **5**: Very lightweight; Python standard library or very small/common pip packages.
- **4**: Lightweight; common libraries such as pandas, numpy, matplotlib, or requests.
- **3**: Medium; specialized but stable pip/conda libraries.
- **2**: Heavy; system dependencies, external tools, complex environments, large downloads, or unstable software.
- **1**: Very heavy; GPU, large local model, paid API, browser automation, complex GUI software, Docker-in-Docker, CUDA, or proprietary service.

Look for complex system dependencies, GPU or large-model requirements, external APIs, paid services, network requirements, large downloads, slow installation, unstable dependencies, and Docker reproducibility issues.

2. `verifiability_score`
This score measures whether the outputs can be automatically tested. Higher is better.
- **5**: Highly structured and easy to check: JSON, CSV, Excel, SQLite, numeric metrics, fixed schema, labels, or validated files.
- **4**: Clear output that can be checked with rules: Markdown report, chart file, table, or formatted file.
- **3**: Partly verifiable, but some subjectivity remains.
- **2**: Mostly natural language and difficult to strictly judge.
- **1**: Highly subjective or creative output; almost impossible to automatically evaluate.

Prefer skills with explicit files, schemas, fields, values, metrics, deterministic outputs, hidden-test potential, or validation procedures.

3. `documentation_quality_score`
This score measures whether the `SKILL.md` is complete, concrete, and reusable. Higher is better.
- **5**: Very complete: clear use cases, inputs, outputs, steps, dependencies, examples, boundary conditions, and error handling.
- **4**: Mostly complete; minor details missing but usable.
- **3**: Understandable, but inputs, outputs, or steps are not detailed enough.
- **2**: Very short or generic; execution requires much guessing.
- **1**: Almost unusable; only a title, one sentence, keyword tags, or vague generic description.

Look for clear applicability, input description, output description, workflow steps, dependencies, edge cases, examples, and non-vague scope.

Important Judging Rules
- Judge only from the provided `SKILL.md` content and skill name.
- Do not reward vague broad capabilities.
- Penalize paid APIs, GPU or large-model inference, browser automation, LibreOffice rendering, OCR with Tesseract, FFmpeg/video-heavy workflows, and complex system components for environment cost.
- Penalize subjective outputs such as "make it better", "creative suggestions", "beautiful essay", and "general advice" for verifiability.
- Return concise reasons grounded in the `SKILL.md`.

Required JSON Shape ...

Table 9: Prompt for evaluating skill suitability for large-scale benchmark task construction.

Graph Composition Evaluation Prompt

You evaluate whether a candidate skill graph substructure can form one real benchmark task.

Output Requirement
Return exactly one JSON object. Do not include markdown.

Input Description
You will receive the following information:
- `structure_type` and difficulty.
- Skills in the candidate.
- `skill_connections`: directed edges such as "skill-a → skill-b".
- Edge evidence: scenario-level handoff examples from source post-scenario to target pre-scenario.
- The full `SKILL.md` content for every skill in the candidate task.

General Judging Rule
Judge the original candidate as a whole. Do not remove, prune, or rewrite the skill set.

Evaluation Dimensions

1. `composition_score`
This score measures whether the given skills can form one task according to the directed structure and order.

This score must consider:
- The task must follow the given direction exactly. For A → B, A's result must be a meaningful input, artifact, state, or prerequisite for B.
- Every skill must have a distinct role. Repeated, redundant, or alternative tools should score low.
- The whole structure must describe one coherent workflow, not a bag of related skills.
- For fan-in, multiple upstream skills must produce complementary inputs for one downstream skill.
- For fan-out, one upstream result must naturally branch into multiple distinct downstream steps.
- For diamond, the split and merge must both be meaningful.
- Edge scores and retrieval evidence are supporting signals only; `SKILL.md` is the source of truth.

Scoring rubric for `composition_score`:
- **5**: The directed structure is clearly necessary and natural; every skill contributes a distinct step; handoffs are explicit.
- **4**: The structure is mostly valid; minor ambiguity, but still a plausible composed task.
- **3**: The skills are related and could be forced into a task, but the order, handoff, or role separation is weak.
- **2**: Mostly topical similarity, wrong direction, repeated or redundant tools, or no real artifact/state handoff.
- **1**: Duplicate or unrelated skills, impossible direction, or no coherent composition.

2. `verifiability_score`
This score measures whether the final task result can be objectively verified for benchmark use.

This score must consider:
- High scores require objective checks on the core success criterion, not only superficial checks.
- Good verification includes exact JSON, CSV, schema, numeric values, labels, deterministic file transformations, code/tests passing, database rows, chart data values, required fields, validated syntax, or rule-checkable evidence.
- Weak verification includes only file existence, generic required sections, keyword presence, or manual visual inspection.
- Low verification includes creative writing, subjective quality, style improvement, coaching, strategy, advice, recommendations, resume quality, presentation quality, design quality, open-ended summaries, or external service state.

Scoring rubric for `verifiability_score`:
- **5**: Core result is highly objective and machine-checkable with strict fields, values, schemas, tests, or deterministic artifacts.
- **4**: Mostly objective and rule-checkable; minor natural language is allowed if the core result is structured.
- **3**: Partly checkable, but important success criteria still require human judgment.
- **2**: Mostly subjective or natural-language output; only superficial checks are possible.
- **1**: Almost entirely subjective, creative, external-service-dependent, or impossible to verify reliably.

Important Judging Rules
- Use `SKILL.md` content as the source of truth.
- Use `skill_connections` and edge evidence only for structure, direction, and handoff evidence.
- Do not reward a task just because each individual skill is useful.
- If the directed composition is weak, `composition_score` must be low.
- If the output is subjective, `verifiability_score` must be low even if the workflow is coherent.
- Do not suggest removing skills. If a skill is redundant, score the original candidate lower.

Required JSON Shape ...

Table 10: Prompt for evaluating whether a candidate skill graph substructure can form a real benchmark task.

Input Entity Fit Evaluation Prompt

You are a benchmark input-entity fit evaluator.

Given a composed task, one upstream skill's full `SKILL.md`, and one candidate input entity snippet, judge whether the entity is suitable as input for that upstream skill in this task.

You only evaluate entity-to-upstream-skill fit. Do not re-evaluate the whole task.

Input Description
You will receive the following information:

1. `task_topology`: skill nodes and directed skill connections in the task.
2. `upstream_skill`: the upstream skill being evaluated, including `skill_id`, `skill_name`, and full `SKILL.md`.
3. `entity`: candidate input entity, including `entity_name`, `entity_type`, `url`, `snippet`, and `why_it_matches`.

Evaluation Aspects
Evaluate only the following three aspects:

1. Topology Match
Judge whether the entity corresponds to an upstream or predecessor skill in the task.

- **Chain**: for A $\rightarrow$ B $\rightarrow$ C, only A's input should be evaluated.
- **Fan-out**: for A $\rightarrow$ B and A $\rightarrow$ C, only A's input should be evaluated.
- **Fan-in**: for A $\rightarrow$ C and B $\rightarrow$ C, evaluate inputs for A and B separately.
- **Diamond**: for A $\rightarrow$ B, A $\rightarrow$ C, B $\rightarrow$ D, and C $\rightarrow$ D, only A's input should be evaluated.
- If the entity is marked for a downstream skill instead of an upstream skill, lower the score.

2. Input Format Match
Use `upstream_skill.skill_md` to infer what input format this skill requires. Judge whether `entity_type` and the snippet's described file or resource type match that requirement.

- If the skill needs CSV and the entity is CSV or tabular data, the format matches.
- If the skill needs VCF and the entity is a general PDF, the format does not match.

3. Input Semantic Match
Use `upstream_skill.skill_md` and `task_topology` to infer what semantic content the upstream skill needs in this task.

- Judge whether the snippet's described content satisfies that semantic need.
- If it is merely in a related domain but cannot serve as the skill input, lower the score.

Do Not Evaluate

- Whether the URL is really downloadable.
- Whether the URL is a homepage.
- Whether the whole task is valid.
- Whether downstream skills can directly use this entity.
- Whether this entity alone is enough to run the entire task.

Scoring Rubric

- **5**: Format and semantics are highly suitable, and the entity targets the correct upstream skill.
- **4**: Mostly suitable, with only minor uncertainty.
- **3**: Somewhat related, but format or semantic fit is clearly uncertain.
- **2**: Weakly related and unlikely to be a valid input for this skill.
- **1**: Not suitable.

Output Requirement
Return exactly one valid JSON object. Do not include markdown.

Required JSON Shape ...

Table 11: Prompt for evaluating whether a candidate input entity fits an upstream skill in a composed benchmark task.

Task Instruction Quality Evaluation Prompt

You are an expert evaluator of task instructions. Your job is to score a given task instruction across several dimensions of clarity, testability, and naturalness.

Evaluation Dimensions
Evaluate the instruction on the following six dimensions, each on a scale from 0 to 10.

1. Goal Clarity
This dimension measures whether the task objective is explicit, concrete, and unambiguous rather than vague.
Common problems include vague action verbs, missing deliverable specification, undefined success criteria, and ambiguous scope.
- **0–2**: Objective is highly vague or unclear.
- **3–4**: Major ambiguity remains.
- **5–6**: Partially clear but still underspecified.
- **7–8**: Mostly clear and actionable.
- **9–10**: Highly specific, concrete, and unambiguous.

2. Input Clarity
This dimension measures whether the required input, source material, system, file, version, or reference is clearly specified.
Common problems include unspecified data source, file, system, or version; ambiguous qualifiers such as "latest"; and unclear input selection.
- **0–2**: Essential input is missing or unclear.
- **3–4**: Major input ambiguity.
- **5–6**: Input is partially defined.
- **7–8**: Input is mostly clear.
- **9–10**: Input is fully and precisely specified.

3. Constraint Completeness
This dimension measures whether the instruction clearly specifies constraints, thresholds, filters, limits, timelines, exclusion criteria, formatting rules, and precision requirements.
Common problems include missing thresholds or limits, missing timelines or deadlines, missing exclusions or filters, and vague precision requirements.
- **0–2**: Constraints are largely missing.
- **3–4**: Many important constraints are absent.
- **5–6**: Some constraints are present but incomplete.
- **7–8**: Constraints are mostly complete.
- **9–10**: Constraints are comprehensive and precise.

4. Referential Clarity
This dimension measures whether all entities, actors, recipients, tools, pronouns, and domain references are clearly identifiable.
Common problems include missing actors or recipients, referent ambiguity such as "it", "they", or "this", and unclear tool, platform, or domain context.
- **0–2**: Severe referential ambiguity.
- **3–4**: Major confusion about actors or references.
- **5–6**: Some ambiguity remains.
- **7–8**: Mostly clear references.
- **9–10**: All references are explicit and unambiguous.

5. Verifiability / Uniqueness of Evaluation
This dimension measures whether the task output can be objectively and uniquely verified, ideally through deterministic checks such as unit tests, exact-match criteria, structured validation, or clearly bounded correctness conditions.
Key principle: high scores require clearly testable correct/incorrect criteria or tightly bounded expected outputs. Low scores correspond to open-ended, subjective, creative, or highly flexible tasks.
- **0–2**: Not objectively verifiable; highly open-ended.
- **3–4**: Weak verification; many acceptable outputs.
- **5–6**: Partially verifiable but still broad.
- **7–8**: Mostly testable with limited ambiguity.
- **9–10**: Strongly and uniquely verifiable with deterministic or near-deterministic evaluation.

6. Human-Likeness of the Instruction
This dimension measures whether the instruction sounds like something a real human would naturally ask, rather than an artificial, awkward, or machine-generated prompt.
Focus on semantic coherence, natural task logic, realistic human intent, and normal phrasing and instruction style.
- **0–2**: Highly unnatural or nonsensical.
- **3–4**: Noticeably artificial.
- **5–6**: Somewhat plausible but awkward.
- **7–8**: Mostly natural and human-like.
- **9–10**: Very natural, realistic, and human-like.

Weighted Scoring
Use the following weights when computing the final weighted average score:
- Goal Clarity: 0.15
- Input Clarity: 0.15
- Constraint Completeness: 0.15
- Referential Clarity: 0.10
- Verifiability / Uniqueness of Evaluation: 0.20
- Human-Likeness of the Instruction: 0.20

Evaluation Guidelines
- Be strict and discriminative in scoring.
- Do not give high scores unless the instruction truly deserves them.
- Penalize any ambiguity that would prevent a model or programmer from executing or validating the task reliably.
- For verifiability, prioritize whether the output can be uniquely checked by code, tests, schemas, exact criteria, or tightly constrained expectations.
- If the task is inherently open-ended, subjective, or admits many reasonable outputs, score verifiability low even if the instruction is otherwise clear.

Output Requirement
Return exactly one valid JSON object. Do not include markdown.
Required JSON Shape ...

Table 12: Prompt for evaluating task instruction quality across clarity, testability, and naturalness dimensions.

Skill Alignment Evaluation Prompt

You are evaluating whether the task instruction naturally requires or can reasonably use every skill bundled with the task.

Input Description
You will receive the following inputs:
- `instruction.md`
- Every `environment/skills/*/skill.md` file

Important Scope
- Evaluate only whether the listed skills are usable for the instruction.
- Do not evaluate tests, solution quality, task value, or missing skills.
- Do not include a `missing_skill_risk` field.

Usage Levels
- **essential**: The task strongly depends on this skill; without it, the task is difficult or unnatural to complete.
- **useful**: The task can be completed without this skill, but this skill is a natural and clearly helpful choice.
- **weak**: The skill has only indirect, marginal, or keyword-level relevance.
- **irrelevant**: The instruction does not naturally call for this skill.

Score Guide for `relevance_score`
- **9–10**: Essential, directly required by the instruction.
- **7–8**: Useful, clearly helpful and naturally triggered.
- **5–6**: Weak, some relation exists but it is not a natural main tool.
- **0–4**: Irrelevant or very forced.

Decision Rules
- **keep**: Every skill has `relevance_score >= 7`.
- **review**: No skill is `<= 4`, but at least one skill is `5-6`.
- **drop_or_fix_skills**: At least one skill has `relevance_score <= 4`.
- **invalid_skill_set**: Use only if skill data is missing or impossible to judge.

Evaluation Rules
- Do not mark a skill useful just because a keyword overlaps with the instruction.
- Do not mark a very broad or general skill useful unless the instruction naturally calls for it.
- If two skills overlap, both may be useful, but explain their distinct roles.
- Evidence must come from the instruction text, not from assumptions about hidden files.
- `matched_skill_capabilities` must come from the `skill.md` content.
- Return each expected skill exactly once.
- Recompute the final decision from the per-skill scores using the rules above.

Output Requirement
Return JSON only, in the exact shape below.

Required JSON Shape ...

Table 13: Prompt for evaluating whether bundled skills are naturally usable for a task instruction.

Test Alignment Evaluation Prompt

Evaluate only whether `tests/test_outputs.py` is aligned with `instruction.md`. Treat `instruction.md` as the source of truth for the task contract. Score every dimension from 0 to 100.

Important Calibration Rules

- Many tasks are fixed-input benchmark tasks. For these tasks, tests may check exact expected values derived from the fixed input files.
- Do not penalize exact numeric or string constants merely because those constants are not written in `instruction.md` or because the input files cannot be inspected in this prompt.
- Treat an assertion as justified if it checks the required output contract or a value plausibly derived from the named task inputs.
- Do not require synthetic edge cases unless `instruction.md` asks for reusable general behavior, a library/API, or tests claim broad generality.
- Anti-cheating is secondary. A fixed expected answer is not automatically weak anti-cheating.
- Penalize anti-cheating mainly when tests only check existence, schema, or loose plausibility without validating substantive results.
- Mark `fatal_issues` only for direct contradictions with `instruction.md`, impossible or unrelated checks, missing or unavailable required inputs, flaky external dependencies, or tests that cannot meaningfully validate the task.

Evaluation Dimensions

- **`alignment_with_instruction`**: Tests validate the required artifact, path, format, and core task goal.
- **`coverage_depth`**: Tests cover key output fields, formats, calculations, and final results.
- **`assertion_justifiability`**: Assertions are justified by `instruction.md` or plausibly by fixed task inputs.
- **`edge_case_coverage`**: Tests cover important boundaries when the task contract calls for general behavior.
- **`determinism_stability`**: Tests avoid flaky randomness, current time, network, or brittle environment state.
- **`anti_cheating_strength`**: Tests validate substantive output rather than only existence, schema, or loose plausibility.
- **`brittleness`**: Reverse item; higher means overstrict exact strings, order, format, or implementation details not required by the task.

Allowed Issue Tags

```
{json.dumps(sorted(ISSUE_TAGS),
    ensure_ascii=False)}
```

Issue Tag Calibration

- Use `test_hardcodes_fixture_answer` only when constants are arbitrary, contradictory, unrelated to fixed inputs, or replace the required task contract with unjustified values.
- Use `test_leaks_expected_answer` only when the test exposes answers in a way that materially weakens validation beyond normal fixed-input ground truth checking.
- Use `hidden_requirement_in_tests` only when a requirement is neither stated in `instruction.md` nor plausibly implied by the fixed inputs and output contract.

`fatal_issues` Format

```
[
  {
    "type": "short_machine_type",
    "description": "brief reason"
  }
]
```

Output Requirement

Return only the following JSON object shape.

Required JSON Shape ...

Table 14: Prompt for evaluating whether task tests are aligned with the instruction contract.