Skill Mining for Agentic AI
A Survey of Methods, Gaps, and a Research Agenda

Anonymous ACL submission

Abstract

LLM agents increasingly store reusable procedures as skills: instructions, workflows, scripts, tool wrappers, and memory records that can be invoked in later tasks. Existing work has made rapid progress on creating and retrieving such artifacts, but the *mining* problem remains under-studied: how should agents discover candidate skills from traces, demonstrations, documents, code, and feedback, and what evidence is needed before those candidates become durable repository entries? This survey formalizes agentic skills as annotated procedural records with discovery metadata, activation conditions, an executable or instructive body, dependencies, validation evidence, and lifecycle state. We define *skill mining* as the data-centric process that maps heterogeneous procedural evidence to repository updates: admit, revise, reject, or retire candidate skills under explicit evidence. We organize recent systems by evidence source, mined object, operator, evaluation signal, and lifecycle operation, connecting skill-centric agent work with data mining, process mining, mining software repositories, lifelong learning, data stream management, and agent memory. Across the literature, reported gains rarely isolate the skill artifact from retrieval policy, executor model, and inference budget. We conclude with a focused research agenda around three questions: what a mined agentic skill should contain, where high-quality mining data can come from, and what a validation–refinement–evolution pipeline should look like.

1 Introduction

LLM agents are beginning to accumulate procedural knowledge. A successful web workflow may become a reusable instruction; a software-engineering trace may become a script; a reflection may become a memory item; a repeated tool pattern may become a named skill. As Figure 1 sketches,

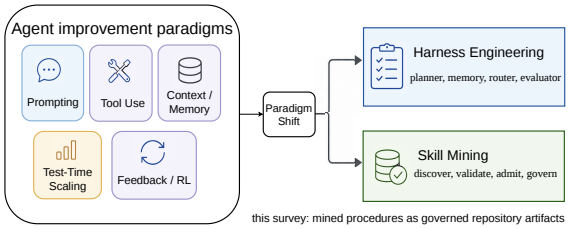

Figure 1: A compact paradigm teaser. Prompting, tool use, context/memory design, test-time computation, and feedback learning improve agent behavior; harness engineering and skill mining are parallel ways to make such improvements persistent, with this survey focusing on mined procedural artifacts.

this shift complements prompting, tool learning, memory, test-time scaling, and feedback learning by asking what reusable procedures should persist beyond a single run. It is visible in Voyager (Wang et al., 2024), Agent Workflow Memory (Wang et al., 2025), Trace2Skill (Ni et al., 2026), Corpus2Skill (Sun et al., 2026), MemSkill (Zhang et al., 2026), SkillOS (Ouyang et al., 2026), and Agentic Skill Discovery (Zhao et al., 2026), as well as in emerging skill package formats from Anthropic (Anthropic, 2025a) and related platforms (Anthropic, 2025b). The community now has many ways to *write*, *generate*, or *retrieve* skills. It has much less agreement on when a candidate skill is sufficiently supported by evidence to be stored, invoked, updated, or removed.

This survey studies that under-developed problem. We use **skill mining** to mean the data-centric process of discovering candidate agentic skills from traces, demonstrations, documents, code, repositories, and feedback, then validating and governing those candidates as durable repository artifacts. The emphasis is deliberately narrower than general agent design. Prompting, planning, tool use, retrieval, and context management shape how an agent behaves at inference time; skill mining asks which observed or authored procedures should per-

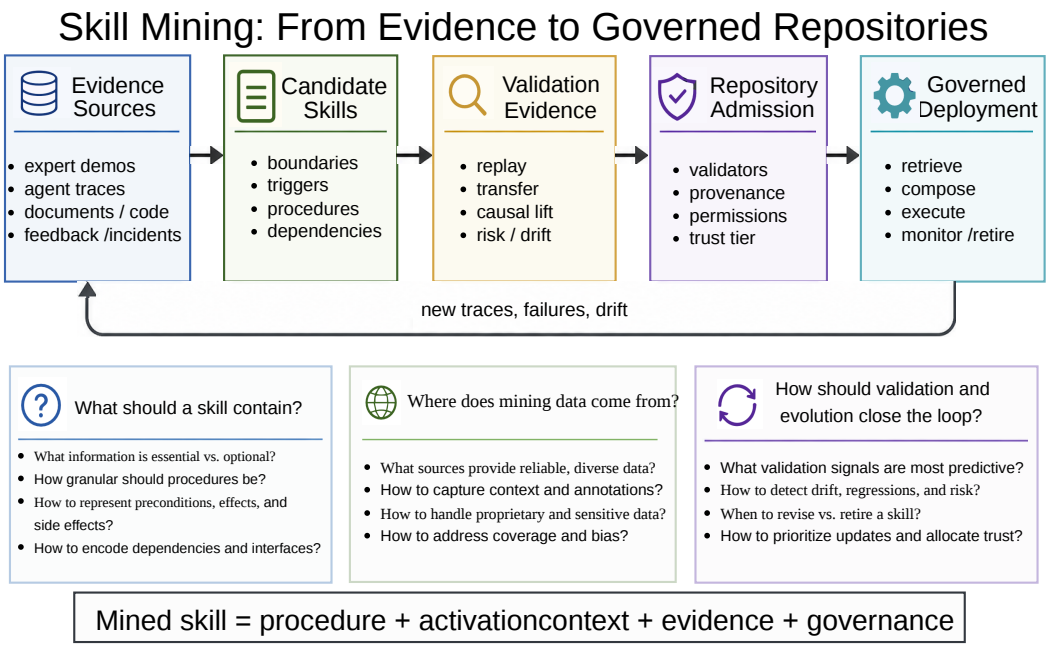

Figure 2: The skill-mining loop used in this survey. The pipeline separates evidence collection, candidate skill objects, validation evidence, and repository lifecycle operations, then connects them to the three research questions in Section 5.

sist across tasks, and under what evidence. A trajectory summary or expert-written instruction is therefore not automatically a mined skill. It becomes one only when the system records its intended activation context, reusable procedure, validation evidence, dependencies, and lifecycle state.

The need for this distinction is already visible empirically. SkillBench reports negative or negligible average gains (Li et al., 2026b); SkillLearnBench shows recursive drift from self-generated skills (Zhong et al., 2026); Agentic Skill Discovery exposes false-positive reward certification (Zhao et al., 2026); and security studies find vulnerable or poisoned community skills (Liu et al., 2026b; Qu et al., 2026). In many papers, final task success entangles the candidate skill, the retrieval policy, the executor model, and the extra inference budget. As a result, it is often unclear whether a stored artifact caused the improvement or merely co-occurred with a stronger agent loop.

This makes skill mining a natural bridge to older data-centric fields. Like data mining, it asks how useful patterns are discovered and validated from data (Fayyad et al., 1996; Han et al., 2011); like process mining, it treats traces as evidence about repeatable procedures and conformance (Van Der Aalst et al., 2003; van der Aalst, 2016); like mining software repositories, it extracts reusable code and API patterns from project histories (Xie and Pei, 2006; Wang et al., 2013). It also inherits lifecycle concerns from lifelong learning and data stream management: environments change, distributions drift, and old knowledge can become harmful if it is not revised or retired. What is new is that a mined skill is operational. It may call tools, access files, issue commands, or change future evidence, so validation and governance are part of the object rather than optional documentation.

This survey makes four contributions. First, it provides formal but lightweight definitions of *agentic skill* and *skill mining*, and separates mining from skill acquisition, retrieval, agent memory, and compound-system optimization. Second, it organizes recent work around the mining decision: what evidence produced a candidate, what object was mined, what admission evidence exists, and how the repository changes over time. Third, it surveys evaluation signals in existing skill-centric and long-horizon agent benchmarks, emphasizing where artifact value is conflated with routing, execution, and extra inference budget. Fourth, it synthesizes a research agenda around three open questions: the

contents of a mined skill record, the sources and curation of high-quality mining data, and closed-loop validation, refinement, and continual evolution.

2 Definitions and Scope

This section defines the paper's object of study. Recent skill surveys organize agent skills broadly around representation, acquisition, retrieval, and evolution (Zhou et al., 2026). We use that work as nearby context, but ask a different question: how a candidate procedure is mined from evidence and admitted into a governed repository. Our definition is therefore grounded in the engineering form that made agent skills practically visible—an agent-loadable package with metadata, instructions, optional resources, and executable support files (Anthropic, 2025b,a; Zhang et al., 2025)—and adds the annotation needed for mining: activation, validation, provenance, and lifecycle state.

2.1 Agentic Skill

Anthropic's Agent Skills design treats a skill as an organized directory: a `SKILL.md` file provides name/description metadata for discovery, the body supplies task-specific guidance, and linked files or scripts are loaded only when needed through progressive disclosure (Anthropic, 2025b,a; Zhang et al., 2025). This engineering pattern is important because it makes procedural knowledge portable, inspectable, and executable without loading everything into the context window.

We use a simple definition. An **agentic skill** is an agent-loadable procedural package that specifies *when* it should be used, *how* to execute the reusable procedure, and *what evidence* supports its use. Formally, we write an annotated skill record as

$$s = \langle m, d, a, b, r, v, \ell \rangle,$$

where m is discovery metadata, d is the domain or task family, a contains activation and abstention conditions, b is the procedural body, r lists resources and permissions, v stores validation evidence, and ℓ records lifecycle state. The key distinction is that skill mining treats validation and lifecycle evidence as part of the object: a plausible procedure remains a candidate until its trigger, behavior, risk, and update state are recorded. This covers executable scripts, workflow memories, memory skills, and hybrid skill graphs (Wang et al., 2024, 2025; Zhang et al., 2026; Liang et al., 2026), while excluding one-off tool calls, raw prompts with no durable identity, and latent model updates with no inspectable repository handle.

2.2 Skill Mining

Skill mining is the data-centric process that turns heterogeneous procedural evidence into governed skill packages. Let $\mathcal{D}$ denote evidence such as agent traces, demonstrations, documents, source code, API histories, user feedback, failures, and incidents; let $\mathcal{S}_t$ be the current skill repository; and let $\mathcal{K}$ collect deployment constraints such as permissions, cost budgets, policy requirements, and tool availability. A mining procedure maps

$$\mathcal{M}(\mathcal{D}, \mathcal{S}_t, \mathcal{K}) \rightarrow (\mathcal{S}_{t+1}, \Delta),$$

where Δ is a set of candidate-level decisions $\delta(c) \in \{\text{ADMIT}, \text{REVISE}, \text{REJECT}, \text{RETIRE}\}$. The output is therefore not merely a list of generated `SKILL.md` files; it is a repository update with evidence explaining each decision.

> **Core decision.** Given a candidate procedure and an existing repository, should this candidate persist as a reusable skill, under what activation conditions, with what risks, and for how long?

This framing keeps the formalism lightweight while adding a mining-specific criterion. A mining system proposes candidates from evidence, packages them so an agent can discover and progressively load them, validates them on held-out or replayable tasks, records risk and permission assumptions, and updates the repository. The key comparison is before/after repository behavior: does adding the candidate improve later tasks beyond what the retrieval policy, executor model, or extra inference budget would already explain?

2.3 Why Mining Is Under-Studied

Current skill-centric work is strongest at producing candidates. Human-authored systems package skills from instructions and workflows, as in Agent Skills (Anthropic, 2025b,a) and Anthropic's engineering guidance (Zhang et al., 2025). Experience-based systems summarize traces into memories or lessons, including Reflexion (Shinn et al., 2023), ExpeL (Zhao et al., 2024), Agent Workflow Memory (Wang et al., 2025), and Trace2Skill (Ni et al., 2026). Programmatic systems synthesize scripts or tools, including Voyager (Wang et al., 2024), CREATOR (Qian et al., 2023), ToolMaker (Cai et al., 2024), SkillCraft (Chen et al., 2026a), and

Table 1: Skill mining vs neighboring problems. The distinctive decision is whether a candidate should persist in a governed repository.

Area	Main question	Relation to skill mining
Skill creation	Can we write or generate a skill?	Produces candidates; mining decides admission and maintenance.
Skill retrieval	Which stored skill should fire?	Assumes admitted skills; mining decides what enters the store.
Agent memory	What should agents remember?	Broader substrate for episodic, semantic, working, and procedural memory; mining focuses on procedural artifacts with validation and lifecycle evidence (Huang et al., 2026).
Compound AI optimization	How should a multi-component system improve?	Optimizes system structure and signals; mining focuses on reusable procedures as repository objects (Lee et al., 2025).
Process/MSR mining	What patterns appear in logs or code?	Provides trace, conformance, and repository-mining tools, but skill mining adds execution, permissions, and drift (van der Aalst, 2016; Xie and Pei, 2006).
Lifelong/data-stream learning	How does knowledge adapt over time?	Supplies drift, forgetting, and streaming perspectives for updating or retiring skills (Parisi et al., 2019; Babcock et al., 2002).

SkillWeaver (Zheng et al., 2025). Routing systems retrieve skills at inference time, including SkillRouter (Zheng et al., 2026), SRA (Su et al., 2026), and SkillRL (Xia et al., 2026). These are necessary components, but they do not by themselves answer the mining question.

The missing part is quantitative admission evidence. SkillsBench reports that self-generated skills do not always improve average performance (Li et al., 2026b); SkillLearnBench shows that continual self-generation can drift (Zhong et al., 2026); Agentic Skill Discovery shows the risk of generated rewards certifying the wrong behavior (Zhao et al., 2026); security studies find vulnerable community skills (Liu et al., 2026b) and poisoned skill ecosystems (Qu et al., 2026). These findings suggest that scaling skill repositories without admission tests can reproduce the brittleness of expert systems: useful rules coexist with stale, over-specific, unsafe, or untriggerable ones (Felgenbaum, 1977).

2.4 Neighboring Problems

Skill mining overlaps with several fields but centers a different decision. Table 1 summarizes the boundary.

2.5 The Skill Mining Loop

Figure 2 gives the high-level loop, while Figure 3 keeps representative work attached to each stage. The field can be organized around four pipeline stages: evidence sources, mining objects, validation evidence, and lifecycle operations. Five separable roles implement this loop: a *source collector* records traces, documents, demonstrations, and incidents; a *miner* proposes bounded procedural candidates; a *validator* tests replay, transfer, causal contribution, and risk; a *repository governor* manages metadata, versions, permissions, trust tiers, and retirement; an *executor* retrieves and applies admitted skills. Separating these roles prevents a common evaluation confound: a system may improve because the executor, router, or model changed, not because the mined skill is better. Section 3 surveys current methods through this pipeline; Section 4 surveys evaluation signals reported or implied by existing benchmarks.

3 The Skill Mining Pipeline in Practice

This section surveys current work through the mining loop introduced in §2.5: each subsection takes one pipeline stage and asks (i) what current systems do at that stage, (ii) what evidence they produce, and (iii) what foundation remains missing. The review is diagnostic. A method may be a strong skill-acquisition algorithm yet still fall short as skill mining if it omits boundary, activation, causal contribution, permissions, or lifecycle evidence. Unlike general skill-lifecycle surveys that organize primarily around representation, acquisition, retrieval, and evolution (Zhou et al., 2026), Table 2 is organized by the mining decision each family informs.

3.1 Evidence Sources: Where Skills Come From

Mining begins with evidence collection. Four sources dominate current practice, each with distinct strengths and failure modes.

Human-derived. Expert SOPs, organizational workflows, demonstrations, and rubrics seed high-precision candidates. Agent Skills (Anthropic, 2025b), AutoGuide (Fu et al., 2024), Anthropic's organization-level guidance (Zhang et al., 2025), and Corpus2Skill (Sun et al., 2026) convert human procedural knowledge into skill packages and directories. Strength: semantic precision—humans encode domain constraints, edge cases, and rare-but-critical handling. Weakness: *tacit knowledge omission*; experts often skip decisive reasoning, leaving the artifact under-specified for novel contexts.

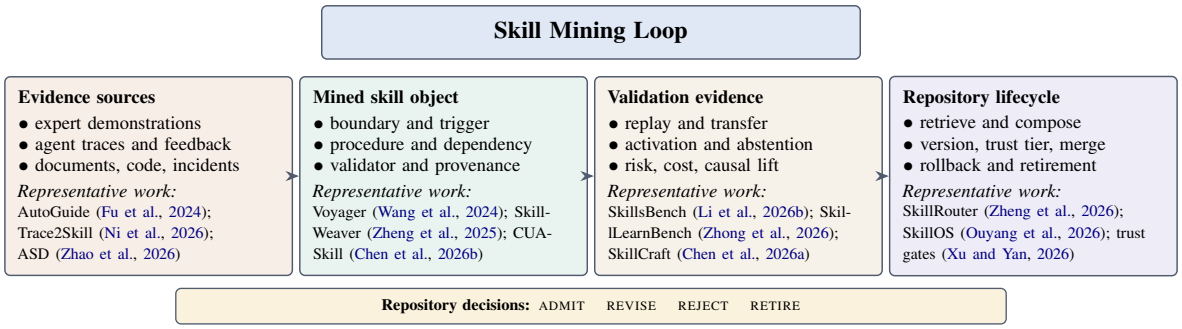

Figure 3: Citation-anchored view of the skill-mining loop. The tree keeps representative work attached to each stage while grouping systems by evidence source, mined object, validation signal, lifecycle operation, and repository decision.

Table 2: Representative method families viewed through skill mining. The rows are grouped by the repository decision each family informs rather than by a general skill lifecycle.

Mining role	Evidence → repository object	Representative anchors	Unsettled admission evidence
Prompt-to-skill externalization	expert docs, SOPs, demos → skill package, guideline, directory	Agent Skills (Zhang et al., 2025; Anthropic, 2025b); AutoGuide (Fu et al., 2024); Corpus2Skill (Sun et al., 2026)	admission tests; anti-trigger coverage; design validation
Experience-to-skill distillation	success/failure traces, feedback → lessons, workflow memory, declarative skills	Reflexion (Shinn et al., 2023); ExpeL (Zhao et al., 2024); AWM (Wang et al., 2025); MemSkill (Zhang et al., 2026); Trace2Skill (Ni et al., 2026)	causal attribution; negative cases; boundary F1; overfit control
Programmatic skill synthesis	code/tool traces, schemas → scripts, APIs, wrappers, graphs	Voyager (Wang et al., 2024); CREATOR (Qian et al., 2023); ToolMaker (Cai et al., 2024); SkillCraft (Chen et al., 2026a); SkillWeaver (Zheng et al., 2025)	typed interfaces; sandboxing; portability; semantic validators
Retrieval, routing, and composition	metadata, state, dependencies → index, activation policy, composition graph	SkillRouter (Zheng et al., 2026); SRA (Su et al., 2026); SkillRL (Xia et al., 2026); CUA-Skill (Chen et al., 2026b); AgentSkillOS (Li et al., 2026a)	activation theory; calibrated abstention; cost/safety/state constraints
Lifelong repository evolution	rewards, failures, drift, validators → insert/update/delete policy, trust tier	ASD (Zhao et al., 2026); AutoSkill (Yang et al., 2026); SkillOS (Ouyang et al., 2026); SkillMOO (Gong et al., 2026); SkillNet (Liang et al., 2026)	update semantics; rollback; staleness diagnosis; feedback control
Benchmarking and risk mining	verifiers, repositories, incidents → utility delta, risk taxonomy, permission card	SkillsBench (Li et al., 2026b); SkillLearnBench (Zhong et al., 2026); Skills-in-the-Wild (Liu et al., 2026b); PoisonedSkills (Qu et al., 2026)	joint utility, safety, maintenance, and trust evaluation

Experience-derived (agent traces). Successful and failed agent trajectories, reflection logs, and recovery sequences. Reflexion (Shinn et al., 2023), ExpeL (Zhao et al., 2024), Agent Workflow Memory (Wang et al., 2025), Trace2Skill (Ni et al., 2026), and MemSkill (Zhang et al., 2026) distill these into lessons, workflow memories, or declarative skills. Strength: aligned with the deployment distribution. Weakness: *self-confirming errors*—an agent's own traces amplify its own mistakes, producing the recursive drift documented in SkillLearnBench (Zhong et al., 2026).

Task-derived (LLM-proposed). An LLM proposes tasks, generates reward and success functions, and bootstraps a skill library from zero. Agentic Skill Discovery (Zhao et al., 2026), CREATOR (Qian et al., 2023), ToolMaker (Cai et al., 2024), and SkillCraft (Chen et al., 2026a) exemplify this pattern in embodied, tool-creation, and code-generation domains. Strength: cold-start capability without human seeding. Weakness: *false-positive admission*—generated rewards can certify behavior falsely, contaminating the library with skills that "work" only under the generated objective.

Corpus-derived. Documents, code repositories, API references, and community knowledge (forums, issue trackers, archives, StackOverflow). Corpus2Skill is a representative attempt to distill enterprise knowledge into navigable agent skills (Sun et al., 2026). Strength: vast scale and real-world distribution. Weakness: *noise, staleness, and unsafe content*; skill-security studies report malicious skills (Liu et al., 2026a), vulnerabilities in community skills (Liu et al., 2026b), and supply-chain poisoning (Qu et al., 2026).

Cross-cutting gap. Across sources, traces are sparse, demonstrations omit latent reasoning, and negative trigger cases are under-sampled, making coverage and overfit control central concerns (Valiant, 1984; Vapnik, 1998; Angluin and Laird, 1988).

3.2 Skill Representations: What Is Mined

The mined object can take several forms, each trading auditability for expressive coverage.

Text-based. SKILL.md packages, declarative procedures, lessons, and memory entries (Reflexion (Shinn et al., 2023), ExpeL (Zhao et al., 2024), AWM (Wang et al., 2025), Trace2Skill (Ni et al., 2026)) are inspectable and portable, but they enlarge the *representation gap*: rule-like content may not capture the latent decision rule.

Code-backed. Scripts, APIs, tool wrappers, and executable subroutines (Voyager (Wang et al., 2024), CREATOR (Qian et al., 2023), ToolMaker (Cai et al., 2024), SkillCraft (Chen et al., 2026a), SkillWeaver (Zheng et al., 2025), CUA-Skill (Chen et al., 2026b)) are executable and deterministic, but semantic correctness beyond passing tests is hard to audit; sandbox and permission risks become first-class concerns.

Memory-based. Episodic and procedural memory, latent skill embeddings, and learned router states are flexible and compact, as shown by MemSkill (Zhang et al., 2026) and the broader agent-memory literature (Huang et al., 2026). Their weakness is inspectability: validators and rollback are harder to define on latent state.

Graph and hybrid. Composition graphs, dependency networks, and hybrid text+code+graph artifacts (PSN (Shi et al., 2026), SkillNet (Liang et al., 2026), AgentSkillOS (Li et al., 2026a)) support structural reasoning over skill dependencies, but they raise annotation cost and make graph maintenance under drift an open problem.

Cross-cutting gap. The *object gap* recurs: many systems produce useful skill content but not the full $\langle m, d, a, b, r, v, \ell \rangle$ record needed for repository audit. A shared representation layer exposing activation, validation, provenance, and lifecycle handles across textual, code, memory, and graph forms is still missing.

3.3 Mining Operators: How Candidates Are Produced

The six operator families in Table 2 are best read as roles in the mining loop: externalization and distillation produce candidates from human or experience evidence; programmatic synthesis turns candidates into executable bodies; retrieval/composition decides when to invoke them; lifelong evolution maintains the repository over time; benchmarking and risk mining audit the result. Each family is methodologically deep but operationally narrow: experience methods often under-specify activation, programmatic methods can hide unsafe dependencies, retrieval methods rarely prove causal contribution, and evolution methods relegate retirement and rollback to second-class status. This is the *identifiability gap*: a system can advance one operator while leaving the rest unmeasured.

3.4 Validation Evidence: What Makes Admission

Once a candidate exists, validation decides whether it deserves repository entry. Four evidence types should travel with every admitted skill.

Replay tests. Executable conformance: insert the candidate into a controlled environment, replay a prefix-preserved trace, and check whether the verifier outcome matches expectation. SkillWeaver (Zheng et al., 2025) and SkillCraft (Chen et al., 2026a) exemplify replay-based self-improvement; process-mining conformance metrics (fitness, precision, generalization) are natural siblings (van der Aalst, 2016).

Causal lift. Counterfactual contribution: the insertion-minus-removal difference under prefix-preserving replay (Pearl, 2009). SkillsBench (Li et al., 2026b) and SkillLearnBench (Zhong et al., 2026) expose negative-lift skills; PoisonedSkills shows that activation alone is insufficient evidence that the skill caused the outcome (Qu et al., 2026).

Activation tests. Positive trigger cases, negative (anti-trigger) cases, and abstention cases. Most systems test positive cases but under-sample negatives, leading to over-firing in deployment. CUA-Skill (Chen et al., 2026b) and SkillRouter (Zheng et al., 2026) expose activation cost and false-trigger penalties.

Risk and drift. Permission violations, unsafe outputs, secret exposure, sandbox failures, and pol-

icy violations, under nominal and adversarial conditions. Skills-in-the-Wild reports 26.1% vulnerability rates (Liu et al., 2026b); PoisonedSkills documents documentation-driven payloads (Qu et al., 2026); Malicious Agent Skills studies broader malicious-skill behavior (Liu et al., 2026a).

3.5 Retrieval and Composition: How Mined Skills Are Used

Admitted skills are useful only if they activate in the right contexts. Current strategies fall into four classes.

Dense embedding over skill names and metadata is common in Voyager (Wang et al., 2024), SAGE (Wang et al., 2026), AutoSkill (Yang et al., 2026), and MemSkill (Zhang et al., 2026); sparse indexes over keywords, symbols, or tool signatures are used by systems such as SkillWeaver (Zheng et al., 2025) and SkillNet (Liang et al., 2026); generative routers produce a skill identifier directly; and structure-aware routers traverse dependencies in SkillRL (Xia et al., 2026), AgentSkillOS (Li et al., 2026a), and Corpus2Skill (Sun et al., 2026). These strategies improve use of a repository, but semantic similarity, generated names, and graph reachability are not by themselves activation evidence.

Cross-cutting gap. Activation theory is missing. None of the families integrates cost, safety, state, and abstention into a unified decision; calibrated abstention—knowing when no admitted skill should fire—remains open.

3.6 Lifecycle Operations: How Repositories Evolve

Admission is the beginning, not the end.

Insert / update / delete. SkillOS learns insert/update/delete curation (Ouyang et al., 2026); SkillMOO performs multi-objective bundle editing (Gong et al., 2026); AutoSkill (Yang et al., 2026) and SkillNet (Liang et al., 2026) provide learned update operators. Update semantics remain under-specified: when does an update become a revision vs. a replacement?

Governance and drift. Xu and Yan's gates and trust tiers map provenance to deployment permissions (Xu and Yan, 2026), while Audited SkillGraph promotes skills through verified experience synthesis (Huang and Huang, 2025). Skill mining places such gates inside the repository loop: static checks, sandbox execution, permission manifests, runtime monitoring, rollback, and retirement become evidence channels. Versioned snapshots and signed provenance mechanisms such as in-toto (Torres-Arias et al., 2019), Sigstore (Newman et al., 2022), SLSA (Open Source Security Foundation, 2025), and SPDX (SPDX Workgroup, 2021) are useful analogues because APIs, GUIs, policies, models, and user preferences drift.

Cross-cutting gap. The *lifecycle gap* is the least developed: retirement, rollback, and drift remain secondary to insertion. A repository should be *versioned as an experimental object*, with explicit incident records, regression tests, and re-validation schedules.

4 Evaluation Landscape

Evaluation in current skill-centric work is fragmented rather than absent. Some benchmarks test whether generated skills improve later tasks; others test long-horizon execution, GUI/tool use, repository curation, or security risk. Read through a mining lens, these benchmarks provide partial signals about four quantities that are often conflated: the mined artifact, the retrieval policy, the executor model, and repository governance. Skill-specific evaluation exposes the issue directly: SkillsBench reports negative deltas on some tasks and no average benefit from self-generated skills (Li et al., 2026b), while SkillLearnBench documents inconsistent gains and recursive drift (Zhong et al., 2026). Long-horizon agent benchmarks place the same attribution problem in tool-rich settings: WebArena (Zhou et al., 2024), OSWorld (Xie et al., 2024), OS-Marathon (Wu et al., 2026), TaskBench (Shen et al., 2024), CUA-Skill (Chen et al., 2026b), SEAgent (Sun et al., 2025), and Agentic Proposing (Jiao et al., 2026). Table 3 organizes these signals descriptively rather than proposing a new benchmark.

Attribution signals. The recurring limitation is attribution. A useful descriptive decomposition is baseline success s_0, success with a newly admitted artifact under a fixed router s_f, and success after router or repository adaptation s_a. The differences $\Delta U_f = s_f - s_0$ and $\Delta U_a = s_a - s_0$, read alongside risk delta ΔR and cost overhead κ, distinguish artifact value from orchestration value. Few current papers report all quantities, but pieces appear across SkillsBench (Li et al., 2026b), SkillLearn-

Table 3: Evaluation signals used or implied by current skill-mining-adjacent benchmarks. Each row corresponds to a repository question rather than to a new benchmark contribution.

Repository question	Observable signal	Anchors and failure mode
Discovery	Trace source, candidate boundary, abstraction level, negative examples, yield, deduplication rate	Trace2Skill (Ni et al., 2026), AWM (Wang et al., 2025): local trace summaries may be mistaken for transferable skills.
Admission	Before/after repository snapshots, held-out tasks, fixed/adapted-router deltas, regression rate	SkillOS (Ouyang et al., 2026), SkillMOO (Gong et al., 2026), SkillNet (Liang et al., 2026): curation and routing gains can hide weak artifacts.
Activation	Positive, negative, and abstention contexts; activation precision/recall; false-trigger cost; retrieval latency	SkillRouter (Zheng et al., 2026), SRA (Su et al., 2026), CUA-Skill (Chen et al., 2026b): a skill may be useful but fire in the wrong contexts.
Execution	Replay harness, verifier, task seeds, budget, insertion/removal, outcome-flip rate	WebArena (Zhou et al., 2024), OSWorld (Xie et al., 2024), TaskBench (Shen et al., 2024), OS-Marathon (Wu et al., 2026): final success may reflect model or budget changes.
Maintenance	Permission manifest, sandbox result, vulnerable-skill tests, API drift, rollback recovery, staleness half-life	Skills-in-the-Wild (Liu et al., 2026b), PoisonedSkills (Qu et al., 2026), SLSA (Open Source Security Foundation, 2025): useful skills can become unsafe or stale.

Bench (Zhong et al., 2026), and SkillOS (Ouyang et al., 2026). Process-mining conformance metrics such as fitness, precision, and generalization are useful analogues (van der Aalst, 2016); skill mining adds causal intervention and risk budgets because the artifact later executes.

5 Open Challenges and Research Agenda

The survey points to many missing pieces, but the most useful research agenda is not a long checklist. We distill it into three questions that define skill mining as a field.

What should an agentic skill consist of? Current systems store different objects: instructions, scripts, memory items, tool wrappers, graph nodes, or package directories. A mining-oriented skill record should specify more than the procedure body. It should annotate discovery metadata, domain, activation and abstention conditions, dependencies, permissions, validators, failure cases, provenance, version, trust tier, and retirement rule. In the notation of Section 2.1, the open problem is to make $\langle m, d, a, b, r, v, \ell \rangle$ concrete enough for tools and benchmarks while flexible enough to cover text, code, memory, and graph skills. This is where our scope differs from broad agent-skill surveys: the central object is a candidate with admission and governance evidence, not only a capability representation.

Where can high-quality mining data come from? The best evidence would be expert action trajectories with reasoning traces: observations, rejected alternatives, tool calls, failure recovery, and success checks. Such data is scarce. Agent traces are easier to collect but inherit agent errors; documents omit tacit decisions; code repositories expose executable patterns but not always task intent. A practical data agenda should combine curated demonstrations, screen recordings, tool logs, issue histories, API examples, and code changes with community knowledge from Reddit, X, Xiaohongshu, StackOverflow, issue trackers, and forums. These sources can reveal real procedures and failures, but they require provenance, privacy review, noise filtering, staleness checks, negative examples, and normalization before they become executable agent behavior.

What should an ideal skill-mining pipeline look like? The ideal pipeline is closed-loop rather than one-shot: collect evidence, propose candidates, package annotated records, validate them with replay and negative-trigger tests, refine failures, admit candidates with sufficient utility and risk evidence, monitor deployment, and revise or retire stale skills. Existing work covers pieces of this loop, but validation, refinement, and continual evolution remain comparatively under-developed; the literature generates more skills faster than it explains when they should be trusted, merged, demoted, or removed. A mature pipeline should use insertion/removal tests, fixed-router and adapted-router comparisons, controlled tool mocks, drift monitors, and risk-adjusted utility $\mathbb{E}[U] - \lambda \mathbb{E}[\mathrm{Risk}]$. Security is part of this loop because malicious skills (Liu et al., 2026a), vulnerable community skills (Liu et al., 2026b), and poisoned repositories (Qu et al., 2026) turn mined procedures into supply-chain artifacts.

6 Conclusion

This survey framed *skill mining* as data-centric discovery, admission, and lifecycle governance for executable procedural knowledge in agentic AI. The central shift is from asking whether agents can generate useful skills to asking when a candidate deserves to persist in a governed repository.

Limitations

This paper is a diagnostic survey and research agenda, not a systematic meta-analysis or an empirical benchmark study. We propose definitions and synthesize an evaluation landscape, but we do not run new experiments or claim exhaustive coverage of all agent-skill systems. The operator taxonomy is contestable, and the quantities in Section 4 are descriptive signals rather than measured results. The proposed schema emphasizes auditable repository handles; latent weight updates and emergent model capabilities may interact with mined skills in ways this paper only partially characterizes.

References

Dana Angluin and Philip Laird. 1988. Learning from noisy examples. *Machine Learning*, 2(4):343–370.

Anthropic. 2025a. Agent skills. https://platform.claude.com/docs/en/agents-and-tools/agent-skills/overview. Accessed 2026-05-14.

Anthropic. 2025b. Agent skills open standard. https://agentskills.io. Accessed 2026-05-14.

Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. 2002. Models and issues in data stream systems. In *Proceedings of the Twenty-First ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '02, page 1–16, New York, NY, USA. Association for Computing Machinery.

Tianle Cai, Xuezhi Wang, Tengyu Ma, Xinyun Chen, and Denny Zhou. 2024. Large language models as tool makers. In *International Conference on Learning Representations*, volume 2024, pages 54067–54089.

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. 2026a. SkillCraft: Can llm agents learn to use tools skillfully? *Preprint*, arXiv:2603.00718.

Tianyi Chen, Yinheng Li, Michael Solodko, Sen Wang, Nan Jiang, Tingyuan Cui, Junheng Hao, Jongwoo Ko, Sara Abdali, Leon Xu, Suzhen Zheng, Hao Fan, Pashmina Cameron, Justin Wagle, and Kazuhito Koishida. 2026b. CUA-Skill: Develop skills for computer using agent. *Preprint*, arXiv:2601.21123.

Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. 1996. From data mining to knowledge discovery in databases. *AI Mag.*, 17(3):37–54.

Edward A. Felgenbaum. 1977. The art of artificial intelligence: Themes and case studies of knowledge engineering. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'77, pages 1014–1029, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Yao Fu, Dong-Ki Kim, Jaekyeom Kim, Sungryull Sohn, Lajanugen Logeswaran, Kyunghoon Bae, and Honglak Lee. 2024. AutoGuide: Automated generation and selection of context-aware guidelines for large language model agents. In *Advances in Neural Information Processing Systems*, volume 37, pages 119919–119948. Curran Associates, Inc.

Jingzhi Gong, Ruizhen Gu, Zhiwei Fei, Yazhuo Cao, Lukas Twist, Alina Geiger, Shuo Han, Dominik Sobania, Federica Sarro, and Jie M. Zhang. 2026. SkillMOO: Multi-objective optimization of agent skills for software engineering. *Preprint*, arXiv:2604.09297.

Jiawei Han, Micheline Kamber, and Jian Pei. 2011. *Data Mining: Concepts and Techniques*, 3 edition. Morgan Kaufmann, Waltham, MA.

Ken Huang and Jerry Huang. 2025. Audited skill-graph self-improvement for agentic LLMs via verifiable rewards, experience synthesis, and continual memory. *Preprint*, arXiv:2512.23760.

Wei-Chieh Huang, Weizhi Zhang, Yueqing Liang, Yuanchen Bei, Yankai Chen, Tao Feng, Xinyu Pan, Zhen Tan, Yu Wang, Tianxin Wei, Shanglin Wu, Ruiyao Xu, Liangwei Yang, Rui Yang, Wooseong Yang, Chin-Yuan Yeh, Hanrong Zhang, Haozhen Zhang, Siqi Zhu, et al. 2026. Rethinking memory mechanisms of foundation agents in the second half: A survey. *Preprint*, arXiv:2602.06052.

Zhengbo Jiao, Shaobo Wang, Zifan Zhang, Xuan Ren, Wei Wang, Bing Zhao, Hu Wei, and Linfeng Zhang. 2026. Agentic proposing: Enhancing large language model reasoning via compositional skill synthesis. *Preprint*, arXiv:2602.03279.

Yu-Ang Lee, Guan-Ting Yi, Mei-Yi Liu, Jui-Chao Lu, Guan-Bo Yang, and Yun-Nung Chen. 2025. Compound AI systems optimization: A survey of methods, challenges, and future directions. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 28760–28775, Suzhou, China. Association for Computational Linguistics.

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

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, et al. 2026b. SkillsBench: Benchmarking how well agent skills work across diverse tasks. *Preprint*, arXiv:2602.12670.

Yuan Liang, Ruobin Zhong, Haoming Xu, Chen Jiang, Yi Zhong, Runnan Fang, Jia-Chen Gu, Shumin Deng, Yunzhi Yao, Mengru Wang, Shuofei Qiao, Xin Xu, Tongtong Wu, Kun Wang, Yang Liu, Zhen Bi, Jungang Lou, Yuchen Eleanor Jiang, Hangcheng Zhu, et al. 2026. SkillNet: Create, evaluate, and connect AI skills. *Preprint*, arXiv:2603.04448.

Yi Liu, Zhihao Chen, Yanjun Zhang, Gelei Deng, Yuekang Li, Jianting Ning, Ying Zhang, and Leo Yu Zhang. 2026a. Malicious agent skills in the wild: A large-scale security empirical study. *Preprint*, arXiv:2602.06547.

Yi Liu, Weizhe Wang, Ruitao Feng, Yao Zhang, Guangquan Xu, Gelei Deng, Yuekang Li, and Leo Zhang. 2026b. Agent skills in the wild: An empirical study of security vulnerabilities at scale. *Preprint*, arXiv:2601.10338.

Zachary Newman, John Speed Meyers, and Santiago Torres-Arias. 2022. Sigstore: Software signing for everybody. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 2353–2367. Association for Computing Machinery.

Jingwei Ni, Yihao Liu, Xinpeng Liu, Yutao Sun, Mengyu Zhou, Pengyu Cheng, Dexin Wang, Erchao Zhao, Xiaoxi Jiang, and Guanjun Jiang. 2026. Trace2Skill: Distill trajectory-local lessons into transferable agent skills. *Preprint*, arXiv:2603.25158.

Open Source Security Foundation. 2025. SLSA: Supply-chain levels for software artifacts specification. `https://slsa.dev/spec/latest/`. Accessed 2026-05-14.

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. 2026. SkillOS: Learning skill curation for self-evolving agents. *Preprint*, arXiv:2605.06614.

German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. 2019. Continual lifelong learning with neural networks: A review. *Neural networks*, 113:54–71.

Judea Pearl. 2009. *Causality: Models, Reasoning, and Inference*, 2 edition. Cambridge University Press, Cambridge, UK.

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

Yubin Qu, Yi Liu, Tongcheng Geng, Gelei Deng, Yuekang Li, Leo Yu Zhang, Ying Zhang, and Lei Ma. 2026. Supply-chain poisoning attacks against LLM coding agent skill ecosystems. *Preprint*, arXiv:2604.03081.

Yongliang Shen, Kaitao Song, Xu Tan, Wenqi Zhang, Kan Ren, Siyu Yuan, Weiming Lu, Dongsheng Li, and Yueting Zhuang. 2024. TaskBench: Benchmarking large language models for task automation. In *Advances in Neural Information Processing Systems*, volume 37, pages 4540–4574. Curran Associates, Inc.

Haochen Shi, Xingdi Yuan, and Bang Liu. 2026. Evolving programmatic skill networks. *Preprint*, arXiv:2601.03509.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pages 8634–8652. Curran Associates, Inc.

SPDX Workgroup. 2021. ISO/IEC 5962:2021 information technology—SPDX specification v2.2.1. `https://spdx.dev/use/specifications/`.

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

Yiqun Sun, Pengfei Wei, and Lawrence B. Hsieh. 2026. Don't retrieve, navigate: Distilling enterprise knowledge into navigable agent skills for QA and RAG. *Preprint*, arXiv:2604.14572.

Zeyi Sun, Ziyu Liu, Yuhang Zang, Yuhang Cao, Xiaoyi Dong, Tong Wu, Dahua Lin, and Jiaqi Wang. 2025. SEAgent: Self-evolving computer use agent with autonomous learning from experience. *Preprint*, arXiv:2508.04700.

Santiago Torres-Arias, Hammad Afzali, Trishank Karthik Kuppusamy, Reza Curtmola, and Justin Cappos. 2019. in-toto: Providing farm-to-table guarantees for bits and bytes. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 1393–1410, Santa Clara, CA. USENIX Association.

L. G. Valiant. 1984. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142.

W. M. P. Van Der Aalst, A. H. M. Ter Hofstede, B. Kiepuszewski, and A. P. Barros. 2003. Workflow patterns. *Distrib. Parallel Databases*, 14(1):5–51.

Wil M. P. van der Aalst. 2016. *Process Mining: Data Science in Action*, 2 edition. Springer, Berlin, Heidelberg.

Vladimir N. Vapnik. 1998. *Statistical Learning Theory*. Wiley, New York.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2024. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*.

Jiongxiao Wang, Qiaojing Yan, Yawei Wang, Yijun Tian, Soumya Smruti Mishra, Zhichao Xu, Megha Gandhi, Panpan Xu, and Lin Lee Cheong. 2026. Reinforcement learning for self-improving agent with skill library. *Preprint*, arXiv:2512.17102.

Jue Wang, Yingnong Dang, Hongyu Zhang, Kai Chen, Tao Xie, and Dongmei Zhang. 2013. Mining succinct and high-coverage API usage patterns from source code. In *Proceedings of the 10th Working Conference on Mining Software Repositories*, pages 319–328. IEEE.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. 2025. Agent workflow memory. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 63897–63911. PMLR.

Jing Wu, Daphne Barretto, Yiye Chen, Nicholas Gydé, Yanan Jian, Yuhang He, and Vibhav Vineet. 2026. OS-Marathon: Benchmarking computer-use agents on long-horizon repetitive tasks. *Preprint*, arXiv:2601.20650.

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. 2026. SkillRL: Evolving agents via recursive skill-augmented reinforcement learning. In *ICLR 2026 Workshop on Lifelong Agents: Learning, Aligning, Evolving*.

Tao Xie and Jian Pei. 2006. MAPO: Mining API usages from open source repositories. In *Proceedings of the 2006 International Workshop on Mining Software Repositories*, pages 54–57. Association for Computing Machinery.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. 2024. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

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

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. 2026. AutoSkill: Experience-driven lifelong learning via skill self-evolution. *Preprint*, arXiv:2603.01145.

Barry Zhang, Keith Lazuka, and Mahesh Murag. 2025. Equipping agents for the real world with agent skills. `https://www.anthropic.com/engineering/equipping-agents-for-the-real-world-with-agent-skills`. Anthropic Engineering Blog, October 16, 2025.

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

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. 2024. ExpeL: Llm agents are experiential learners. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(17):19632–19642.

Xufeng Zhao, Cornelius Weber, and Stefan Wermter. 2026. Agentic skill discovery. *Robotics and Autonomous Systems*, 196:105248.

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

YanZhao Zheng, ZhenTao Zhang, Chao Ma, YuanQiang Yu, JiHuai Zhu, Yong Wu, Tianze Xu, Baohua Dong, Hangcheng Zhu, Ruohui Huang, and Gang Yu. 2026. SkillRouter: Skill routing for llm agents at scale. *Preprint*, arXiv: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. 2026. SkillLearnBench: Benchmarking continual learning methods for agent skill generation on real-world tasks. *Preprint*, arXiv:2604.20087.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. 2024. WebArena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, volume 2024, pages 15585–15606.

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

A Statement on LLM Usage

The authors used large language models as research and writing assistants. The assistance included deep literature search support for identifying potentially relevant papers, organizing and comparing related work, drafting and polishing prose, and proposing figure layouts. All cited papers, claims, definitions, and final writing decisions were manually checked and revised by the authors. The models were not used as autonomous scientific agents, and the authors remain responsible for the paper's content, framing, and conclusions.