SkillSeek: Plug-and-Play Skill Retrieval for Open-Source Agentic Workflows

Guanqun Yang
Stevens Institute of Technology
Hoboken, NJ, USA
gyang16@stevens.edu

Abstract

Anthropic Agent Skills package reusable procedural know-how into `SKILL.md` files, but extracting their value at scale requires a curator who reads the pool and picks a small topical bundle for each task. Existing benchmarks score skills as part of a fixed (model, harness) bundle, which leaves the retrieval sub-problem entangled with model and harness choices. We isolate that sub-problem in an open-source, specification-faithful stack (the OpenHands SDK, which implements the AgentSkills specification end-to-end, driving `gpt-oss-120b`) and ask how much of the hand-curated gain a deterministic plug-and-play retriever can recover. SkillSeek is a CPU-only two-stage retriever (BGE-base bi-encoder followed by a `bge-reranker-v2-m3` cross-encoder, top-20 to top-5) exposed as a Model Context Protocol (MCP) server. On the 89-task SkillsBench [11], SkillSeek reaches 0.467 task-mean pass rate, within 1.7% of the per-task hand-curated `oracle` bundle (0.484), +8.3% over no-skills (0.384), and +8.0% over a load-everything baseline (0.387); on easy tasks SkillSeek outperforms `oracle`. The cost is bounded at 1.3 s of CPU-only retrieval latency per call and +15% tokens over no-skills, with no API spend and no end-to-end slowdown. Together, these results suggest that hand-curated skill selection in deployed agent systems can be largely replaced by off-the-shelf retrieval at modest cost. https://anonymous.4open.science/r/SkillSeek/

ACM Reference Format:
Guanqun Yang. 2026. SkillSeek: Plug-and-Play Skill Retrieval for Open-Source Agentic Workflows. In *Proceedings of Conference Name (Conference).* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

Anthropic released the Agent Skills specification in October 2025, packaging procedural know-how for an LLM agent into `SKILL.md` files that load progressively at startup, on activation, and on demand [15].[1] The ecosystem has grown rapidly: Li et al. [11] aggregate 47,150 unique skills from open-source repositories and corporate partners; Gao et al. [4] crawl 55,315; Chen et al. [2] index 118,000. At this scale, the cost of authoring skills is dwarfed by the cost of *selecting* them: the canonical evaluation in SkillsBench requires the task author to read the catalog and hand-pick a small bundle for each task [11], and recent empirical work shows that mounting more skills without selection regresses to no-skills behavior [13].

How well skills work in an agent loop is therefore a question about both the model's ability to follow procedural guidance and the harness's ability to surface the right guidance at the right moment. Li et al. [11] characterize this jointly, evaluating seven *model+harness* configurations as fixed pairs (Claude Code with each of Opus 4.5, Sonnet 4.5, and Haiku 4.5; Codex CLI with GPT-5.2; Gemini CLI with Gemini 3 Pro and Gemini 3 Flash) and reporting an aggregate +16.2% pass-rate gain across them. This bundled measurement is informative as a first pass and a fair reflection of how agents are deployed today, but it leaves open how much of the gain attributes to a particular model family, how much to a particular harness's skill-discovery loop, and which subcomponent a developer should change first when moving to a different stack.

A complementary line of work begins to factor these choices. Chen et al. [2] address *cross-(model, harness) portability* of skill execution, treating skills as code compiled per-target across 8 LLMs and 3 harnesses. Liu et al. [13] address *realistic skill discovery* over a 34,000-skill pool with agentic-hybrid retrieval and query-specific refinement, but still pair each model with its native CLI. Concurrent with the present work, that study delegates second-stage candidate filtering to the agent itself rather than to a cross-encoder reranker. Neither isolates the retrieval-into-an-open-source-harness setting that many developers face: an open-source agent runtime (e.g., OpenHands), an open-weight model that does not have a vendor-built CLI, and a community pool of `SKILL.md` packages whose per-task curation is impractical at scale.

This paper takes that modular view. We hold the harness fixed (OpenHands SDK, with skills served over MCP) and the model fixed (`gpt-oss-120b`, served locally via vLLM), and ask how much of the effectiveness gain reported for hand-curated skill bundles a deterministic plug-and-play retriever can recover, over what pool size, and at what cost in latency, tokens, and ranking precision. We instantiate this retriever as SkillSeek, a CPU-only two-stage system

[1] The open spec is hosted at https://agentskills.io/specification.

Conference, City, Country
2026. ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

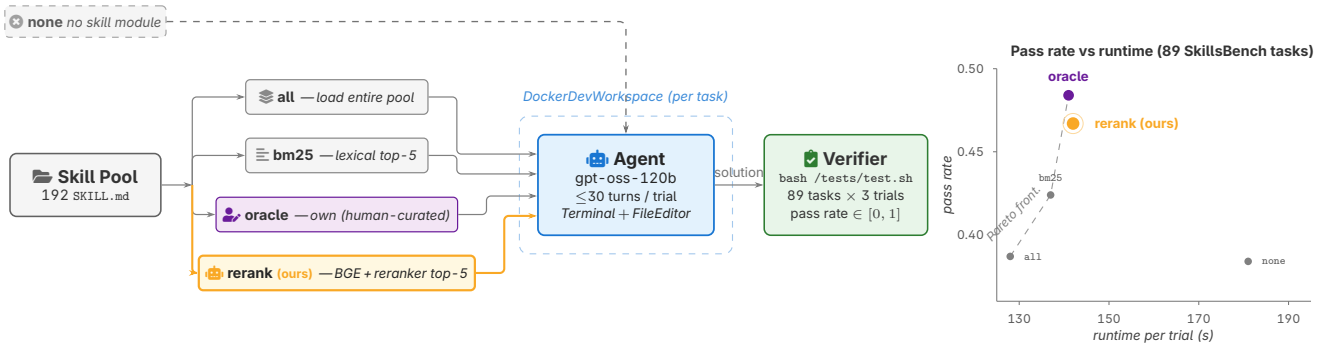

Figure 1. Overview of SkillSeek. The agent consults the skill pool through one of five conditions: `none` (bypass), `all` (load every skill), `bm25` (lexical top-5), `oracle` (per-task hand-curated bundle), or **`rerank`** (SkillSeek: BGE bi-encoder followed by a cross-encoder, top-5). Pass rate is computed by a deterministic pytest verifier per task. Right: task-mean pass rate vs. per-trial runtime; SkillSeek sits on the Pareto front.

(BGE-base bi-encoder followed by a `bge-reranker-v2-m3` cross-encoder, top-20 to top-5) exposed as an MCP server. The agent's normal skill-discovery loop calls `skill_lookup` to obtain a top-5 candidate set and `skill_load` to fetch the chosen skill's body; the retriever sits behind these calls and is interchangeable across LLMs and agent harnesses that already speak MCP. Figure 1 depicts the deployment, contrasting SkillSeek against the four baseline conditions (`none`, `all`, `bm25`, `oracle`) used in our evaluation. We discuss the choice of OpenHands among candidate harnesses in §2.3.

Contributions. Our contributions are as follows:

- **System.** We present SkillSeek, a CPU-only two-stage retriever (BGE-base bi-encoder followed by a `bge-reranker-v2-m3` cross-encoder, top-20 to top-5) exposed as an MCP server, that recovers near-oracle task pass rate without any per-task human curation.
- **Empirical evaluation.** We evaluate SkillSeek on the 89-task SkillsBench [11] and find a task-mean pass rate of 0.467, within 1.7% of the per-task hand-curated `oracle` bundle (0.484) and +8.3% over the no-skills baseline (0.384); on the 6 easy tasks, SkillSeek (0.953) outperforms even `oracle` (0.877) (§3.2).
- **Architecture analysis.** We compare seven retriever architectures and find that a 568 M-parameter cross-encoder reranker outperforms every larger bi-encoder we tested on this benchmark, including BGE-large (335 M) and Qwen3-Embedding at 0.6 B/4 B/8 B (§3.3).
- **Cost characterization.** We measure that plugging SkillSeek into the agent loop adds only 1.3 s of CPU-only retrieval latency per call and +15% tokens over no-skills, with no API spend and no end-to-end slowdown (§3.4).

2 Method

SkillSeek is a deterministic two-stage retriever that sits between the agent and the skill pool, exposed as an MCP server so that it integrates with any harness that already speaks MCP. The system overview is depicted in Figure 1.

2.1 Two-Stage Retriever

Given a skill pool, SkillSeek indexes each skill by the text "`<name>:<description>`" extracted from its `SKILL.md` YAML frontmatter; the body of the skill is not indexed and is fetched only on demand, preserving the progressive-disclosure layering of the AgentSkills specification [15]. At lookup time the agent submits a natural-language query. The first stage scores the query against every indexed skill with the BGE-base bi-encoder (110 M parameters, 768-dim embeddings) and returns the top-20 candidates by cosine similarity. The second stage reranks the 20 candidate (query, skill) pairs jointly with the `bge-reranker-v2-m3` cross-encoder (458 M parameters) and returns the top-5 by cross-encoder score. We deliberately do not prepend an instruction template to the query, even though `Qwen3-Embedding`'s documented inference format requires one [16], so that the comparison across retrievers in §3.3 is identical at the input layer. The choice of architecture follows the empirical observation that bidirectional encoders with a `[CLS]`-as-aggregator inductive bias are well-suited to short tag-like text [1], where decoder-only embedders trained for long-context retrieval require explicit adaptation to compete [10].

2.2 MCP Server Interface

We expose the retriever as an MCP server rather than as a baked-in component of the agent runtime so that any harness that already speaks MCP can adopt SkillSeek without source-level changes; the same server can be pointed at by Claude Code, Codex CLI, OpenHands, or other MCP-aware harnesses with no modification, and we choose OpenHands as our experimental harness for reasons discussed in §2.3. SkillSeek exposes three tools to the agent over MCP. `skill_lookup(query,k)` returns the top-k candidates as "`<name`

>(score=*s*):<description>" lines. skill_load(name) returns the full SKILL.md body for a named skill. skill_list() returns every skill name and description; the agent rarely uses it, but it is useful for manual debugging. A server-side X-Skill-Method HTTP header lets the experimental driver override which retriever skill_lookup runs; the agent always sees the same tool schema regardless of which retriever is in use, keeping the comparison clean across the conditions of §3.

2.3 Driver and Harness

Choice of agent harness. Our retrieval conditions (bm25, rerank) are exposed as MCP servers, so the harness must (i) speak MCP natively and (ii) be driveable from Python with the open-weight gpt-oss-120b backbone we use; closed harnesses or harnesses lacking either property would either prevent the experiment from running or introduce harness-side confounders that no retriever-side ablation can remove. Five candidate harnesses were available at the time of the experiments:

- **Claude Code** (Anthropic) is closed-source: its skill-discovery loop is unauditable, so we cannot factor harness-side optimizations out of any reported gain.
- **Codex CLI** (OpenAI) is closed-source for the same reason and additionally tied to a single model family.
- **Gemini CLI** (Google) is open-source but ships only as a Node.js subprocess controlled over stdio with no Python SDK, and is tied to the Gemini model family, so we cannot run our open-weight gpt-oss-120b backbone through it.
- **Terminus-2** (used by SkillsBench evaluations and by Liu et al. [13] for Kimi K2.5) is a deliberately-minimal benchmark baseline (∼1.5 K LOC, no native MCP, no skill module); SkillsBench added skill-loading via a ∼1.1 K-LOC fork but MCP was never wired in, so our bm25/rerank conditions cannot run on it without us forking it ourselves.[2]
- **OpenHands SDK** (All Hands AI) is open-source, model-agnostic, Python-driveable, and ships first-class MCP and skill modules.

We therefore use the OpenHands SDK: among the candidates above, it is the only one combining an in-process Python agent loop with first-class MCP, which our experimental driver depends on for per-turn event capture and direct mcp_config injection.[3]

Driving the OpenHands SDK.. We drive the OpenHands SDK directly rather than through the upstream openhands-acp CLI to avoid an ACP↔MCP schema mismatch in the version available at the time of the experiments; the workaround is documented in Appendix A. Skills are not bulk-mounted into the agent's initial context; instead, the agent calls skill_lookup on demand, which keeps the system-prompt overhead independent of pool size. For the none, oracle, and all conditions of §3, the agent never calls MCP and the corresponding pre-mounted skill set is supplied through the harness's standard skill-mounting path. For the retrieval conditions (bm25, rerank), we append a one-paragraph system-message suffix instructing the agent to call skill_lookup before exploring the workspace; without this prompt, gpt-oss-120b ignores the MCP tools entirely. The suffix is added only when an MCP URL is configured, so the none/oracle/all conditions are unaffected.

3 Experiments

This section answers three research questions:

RQ1 (Effectiveness). Does retrieval over the full pool recover the per-task oracle's task pass rate, and where does it still fall short? (§3.2)

RQ2 (Architecture). Can any alternative retriever architecture beat the one SkillSeek uses, when tried on a development subset? (§3.3)

RQ3 (Cost). What runtime and token overhead does plugging the retriever into the agent loop add? (§3.4)

3.1 Experimental Setup

We evaluated SkillSeek on the 89-task SkillsBench [11] under the following five conditions:

- none: mounts no skills.
- oracle: mounts the per-task hand-curated bundle of 1 to 7 topical skills (all relevant).
- all: mounts the deduplicated 192-skill pool, capped at 150 by the agent loader (Appendix B).
- bm25: mounts the lexical top-5.
- rerank: mounts SkillSeek's top-5 from the two-stage neural retriever.

Every task was run for 3 trials per condition, yielding $5 \times 89 \times 3 = 1{,}335$ trials. We chose gpt-oss-120b as the agent backbone for two reasons: at the time of the experiments, it was the most capable open-weight LLM[4] with (i) a mature vLLM serving stack and (ii) a memory footprint that fit our 2× RTX 6000 testbed (48 GB each); evaluating the more recent google/gemma-4-31b-it on the same sweep is left as future work. The agent harness is the OpenHands SDK, chosen and motivated in §2.3. Appendix A documents

[2] Independently, Liu et al. [13] write in their Table 2 footnote (p. 8): *"Kimi's query-agnostic results are omitted because Terminus-2 does not support sub-agent operations that are needed."*

[3] Qwen-Code (Alibaba, derived from Google Gemini CLI per its README; used by Liu et al. [13] for Qwen3 backbones) is a comparably-capable open-source alternative that supports OpenAI-compatible model endpoints and ships native MCP, AgentSkills-spec skill loading, and subagents; it is a natural target for a future-work reproduction of our results.

[4] https://llm-stats.com/leaderboards/open-llm-leaderboard.

Condition	Task-mean
none	0.384
all	0.387
bm25	0.424
SkillSeek (rerank)	**0.467**
oracle	0.484

Table 1. Per-condition results on the 89-task SkillsBench (5 conditions × 89 tasks × 3 trials = 1,335 trials, agent backbone gpt-oss-120b). oracle mounts the hand-curated bundle of 1–7 topical skills per task; all mounts the 150-skill pool, the bulk of which are distractors for any given task. Conditions are sorted by ascending task-mean pass rate.

the agent harness, the serving stack, and the per-trial caps used to keep the sweep tractable.

3.2 RQ1: Effectiveness vs. Hand-Curation

Table 1 reports the main result. SkillSeek (rerank) attains a task-mean pass rate of 0.467, within 1.7% of the per-task hand-curated oracle bundle (0.484) and +8.3% over none (0.384). The load-everything baseline all (0.387) is statistically indistinguishable from none: mounting the entire 150-skill pool swamps the relevant entries with distractors and gives the agent nothing usable that selection has not already provided. SkillsBench itself evaluates only none, oracle, and self-generated skills [11]; we add the all baseline to test whether mere exposure to the pool substitutes for selection. Selection turns out to do almost all of the work: oracle mounts on the order of 2 to 3 topical skills without distractors; SkillSeek mounts a top-5 retrieval that is mostly relevant without per-task human curation; all mounts the same relevant skills as oracle *plus* roughly 145 distractors and recovers no benefit.

oracle *is a topical oracle, not a strict one.* oracle's ceiling of 0.484 is bounded above by the curator's attention budget rather than by the best possible per-task choice from the pool. Stratifying the 55 tasks with 1 to 5 skill bundles (the remaining 34 tasks either have larger bundles or have no successful oracle trials, leaving too few per bucket to stratify reliably) gives lift over none of −0.01 at size 1 (22 tasks), +0.16 at size 2 (12 tasks), +0.18 at size 3 (10 tasks), +0.10 at size 4 (5 tasks), and −0.24 at size 5 (6 tasks); Pearson r between bundle size and oracle task-mean is −0.089. The 5-skill bucket is the decisive case: even when every individual skill is topically related, bundles of that size induce context dilution and attention confusion. We accordingly read oracle as a *topical* oracle (every skill is domain-related, enforced at PR review) rather than a *strict* oracle (which would require monotone benefit with more skills), so the 1.7% gap between SkillSeek and oracle is a gap to a curator-with-attention-floor rather than to a per-task optimum.

Difficulty	n tasks	none	all	bm25	**rerank**	oracle
easy	6	0.851	0.792	0.879	**0.953**	0.877
medium	52	0.391	0.449	0.476	**0.504**	0.558
hard	26	0.387	0.280	0.372	**0.323**	0.312

Table 2. Per-condition task-mean pass rate stratified by SkillsBench's task-difficulty label. 5 of 89 tasks lack a difficulty label and are excluded.

Stratification by task difficulty. Table 2 breaks down the headline result by SkillsBench's task-difficulty label. Three patterns stand out. On the 6 *easy* tasks, SkillSeek (0.953) outperforms even the curated oracle bundle (0.877) and the load-everything all baseline is the only condition that hurts. The bulk of the 1.7% gap to oracle accumulates on the 52 *medium* tasks, where oracle (0.558) leads, SkillSeek (0.504) is second, and all (0.449) is essentially tied with none (0.391). On the 26 *hard* tasks, *no* condition reliably helps: none (0.387) edges out both oracle (0.312) and SkillSeek (0.323); the agent appears to be in the regime where additional skills compete with reasoning budget rather than substitute for missing knowledge.

Where does retrieval still fall short? The remaining 1.7% gap to oracle is a *ranking* failure, not a *coverage* failure: every gold skill in an oracle bundle is by construction in the 192-skill pool that SkillSeek retrieves over (the pool is the union of per-task oracle bundles, deduplicated; see Appendix B), so when SkillSeek loses to oracle, the gold skill is present in the retriever's index but ranked below a topically similar near-twin. We diagnose this gap through two complementary analyses, both adapted from the citation-quality framing of Chen et al. [3], which reports precision and recall of agent-cited documents against gold evidence. We extend that framing to the two retrieval stages of our harness: the top-k set L_t returned by skill_lookup on task t, and the subset $D_t \subseteq L_t$ that the agent then fetched via skill_load. Let G_t denote the gold oracle bundle for task t. We define

$$P_{\text{lookup}} = \frac{|L_t \cap G_t|}{|L_t|}, \qquad R_{\text{lookup}} = \frac{|L_t \cap G_t|}{|G_t|},$$
$$P_{\text{load}} = \frac{|D_t \cap G_t|}{|D_t|}, \qquad R_{\text{load}} = \frac{|D_t \cap G_t|}{|G_t|}.$$

Analysis 1: Stage-wise citation precision/recall. Table 3 confirms that the cross-encoder does real ranking work: SkillSeek attains $P_{\text{lookup}} = 0.346$, the highest of any method tested and substantially above the 0.244 to 0.286 band of bi-encoder configurations. A subtler observation: the load-stage precision P_{load} does not perfectly track task-mean across conditions, with embed and embed_large achieving $P_{\text{load}} = 1.000$ yet underperforming rerank on end-to-end task pass rate. The reason is specific to our harness rather than the AgentSkills specification: the agent makes about 1.0 skill_load call per trial (Table 6), so it must rely on *how*

Condition	n trials	P_{lookup}	R_{lookup}	P_{load}	R_{load}
bm25	215	0.286	0.681	0.920	0.448
embed (BGE-base)	24	0.244	0.717	1.000	0.532
embed_large (BGE-large)	23	0.274	0.801	1.000	0.564
qwen0_6b	24	0.211	0.629	0.842	0.487
qwen4b	19	0.282	0.800	0.867	0.523
qwen8b	24	0.181	0.717	0.875	0.514
SkillSeek (rerank)	**212**	**0.346**	0.759	0.930	0.469

Table 3. Citation precision and recall per retrieval condition, summed over the five retrieval-condition methods ($n = 541$ trials total). Metrics are defined in the body.

Task	oracle	rerank	Δ
SkillSeek wins (top 5)			
bike-rebalance	0.45	0.85	+0.40
3d-scan-calc	0.67	1.00	+0.33
mario-coin-counting	0.67	1.00	+0.33
dialogue-parser	0.77	1.00	+0.23
dapt-intrusion-detection	0.72	0.91	+0.19
SkillSeek loses (top 5)			
pdf-excel-diff	1.00	0.55	−0.45
citation-check	0.78	0.33	−0.45
energy-ac-optimal-power-flow	0.43	0.00	−0.43
grid-dispatch-operator	0.67	0.25	−0.42
taxonomy-tree-merge	0.57	0.21	−0.36

Table 4. Per-task wins and losses for SkillSeek (rerank) vs. oracle on the 89-task sweep. Top 5 in each direction.

the retriever ranks its top-5, not just on whether the gold skill appears anywhere in the top-5.[5]

***Analysis 2: Per-task delta against* oracle.** Table 4 reports the largest per-task differences. SkillSeek wins on tasks where the agent's lookup query and the gold skill's name+description are textually close, so the cross-encoder's joint scoring locks onto the gold skill (bike-rebalance, mario-coin-counting, dialogue-parser). It loses on tasks where the gold skill encodes a domain-specific schema or arithmetic procedure and another pool skill is its near-twin in description (pdf-excel-diff, energy-ac-optimal-power-flow, taxonomy-tree-merge). For example, on pdf-excel-diff, the gold pdf-form-extraction skill is in the pool but a topically-overlapping spreadsheet-diff skill is ranked higher; both look on-topic from the lookup query, only one matches the verifier's exact format. The cause is not that the right skill is missing or that the retriever model is generally weak; rather, for these queries the candidate pool contains multiple lexically and semantically similar entries, and the cross-encoder lacks the verifier-aware signal needed to discriminate between them.

[5]The AgentSkills specification [15] defines progressive disclosure but does not prescribe a discovery interface or ranking semantics; both are left to the host harness.

Condition	Params	Task-mean	n trials	Served from
qwen0_6b†	600 M	0.609	20	local CPU
qwen4b	4 B	0.640	20	OpenRouter
qwen8b	8 B	0.682	22	OpenRouter
embed (BGE-base)	110 M	0.702	14	local CPU
embed_large (BGE-large)	335 M	0.703	20	local CPU
SkillSeek (rerank)	**568 M**	**0.736**	20	local CPU

Table 5. Method-selection ablations on the 10-task development split. rerank is BGE-base (110 M) followed by bge-reranker-v2-m3 (458 M cross-encoder) over the top-20 candidates; the parameter count 568 M reflects the sum. The five Qwen / BGE bi-encoder rows have full preserved trial logs; the rerank row reports the original method-selection-time task-mean (the per-trial logs for these conditions were overwritten by the later full-corpus sweep, see Appendix B). † qwen0_6b is the same Qwen3-Embedding family as qwen4b/qwen8b but served locally via sentence-transformers on CPU rather than through OpenRouter; we include it to rule out an API-quality explanation for the family's underperformance.

Takeaway. SkillSeek reaches near-oracle quality (within 1.7%) at zero per-task curation cost; the residual gap is a ranking failure on tasks whose pool contains multiple near-twin skills indistinguishable from the lookup query.

3.3 RQ2: Retrieval Architecture

Given that the cross-encoder reranker recovers near-oracle quality at the 89-task scale (§3.2), a natural follow-up is whether a different retriever architecture can do better. We took a 10-task development subset and swapped SkillSeek's BGE-base bi-encoder for two alternatives: scaling the same family (BGE-large, 335 M; 3× the parameter count) and switching family to Qwen3-Embedding at 0.6 B, 4 B, and 8 B. Table 5 reports the result. Neither move beats the cross-encoder reranker. Scaling the bi-encoder gives no measurable lift: BGE-large (0.703) ties BGE-base (0.702). The entire Qwen3-Embedding family underperforms BGE-base across all three sizes (0.609/0.640/0.682 for 0.6B/4B/8B) even though Qwen models top general-purpose retrieval leaderboards[6]. The cross-encoder reranker remains the best configuration on the same split at 0.736.

Three factors explain why a smaller BGE-base outperforms larger Qwen3-Embedding models in our setting.

- **Architecture.** BGE-base is built on the bidirectional XLM-RoBERTa encoder, with the [CLS] hidden state trained as a global aggregator of the input by construction [1]. Decoder-only LLMs of the kind Qwen3-Embedding builds on encode the last-token hidden state for local next-token prediction; turning this into

[6]https://huggingface.co/spaces/mteb/leaderboard

Condition	MCP latency/trial	Wall clock	Tokens vs. none
none	0 ms	181 s	268 K
oracle	0 ms	141 s (−40)	314 K (+17%)
all	0 ms	128 s (−54)	426 K (+59%)
bm25	0 ms	137 s (−45)	301 K (+12%)
SkillSeek (rerank)	**1,342 ms**	142 s (−39)	308 K (+15%)

Table 6. Per-trial overhead on the full 89-task sweep. "MCP latency/trial" sums the wall-clock cost of all skill_lookup and skill_load calls within a trial. "Wall clock" and "Tokens vs. none" are reported as deltas relative to the none row.

a global summary suitable for retrieval requires explicit adaptation [10]. On short tag-like inputs, where there is no later context for a causal model to summarize, the bidirectional [CLS] aggregator is structurally well-suited.

- **Training distribution.** Zhang et al. [16] train Qwen3-Embedding on ∼150 M foundation-model-synthesized pairs spanning multilingual, long-document, and code retrieval; the BGE family's training mix includes MS MARCO, NQ, HotpotQA, and MIRACL plus weakly-supervised Q&A-forum and academic-paper pairs [1]. Our short name:description index is closer in granularity and form to the BGE training distribution.
- **Corpus scale.** The gold skill is already in the top-5 for most queries (Table 3: $R_{\text{lookup}} = 0.759$ for SkillSeek, 0.629–0.801 across all retrievers), so what differentiates retrievers in this regime is *ranking discrimination* (whether the gold skill is at position 1 rather than position 5), not *recall* (whether the gold skill appears anywhere in the top-5). Larger embedders trained on million-document or multilingual corpora primarily improve recall over noisy candidate pools, leaving little room to express that advantage on our 192-skill index.

We separately rule out an API-quality explanation: qwen0_6b, run locally via sentence-transformers on CPU, remains the worst-performing retrieval condition tested.

Takeaway. The cross-encoder reranker, not embedder scale, is what closes the gap to oracle-level retrieval.

3.4 RQ3: Runtime and Token Cost

Table 6 shows that the runtime cost of plugging SkillSeek into the agent loop is dominated by retrieval inference: a median 1.1 s per skill_lookup call, with about 1.1 lookup and 1 load call per trial. The cumulative MCP latency per trial is therefore 1,342 ms.

This latency is not a tax: the rerank condition finishes 39 s faster on average than none, and every skill-using condition is faster than none. The mechanism is straightforward: without skills the agent flounders, taking more turns, more retries, and more 600 s timeouts; with the right skill loaded early the trial converges on the first or second attempt. The 1.3 s of MCP overhead pays for itself many times over in saved agent-loop time. Token cost is similarly modest: rerank uses +15% tokens over none, compared to +59% for all, which inflates the system prompt with 150 skill descriptions.

Method	*n* calls	Median (ms)	p95 (ms)
skill_load	584	0.02	0.03
bm25	296	0.3	0.6
embed (BGE-base)	30	17	6,317
qwen0_6b†	27	52	82
embed_large (BGE-large)	25	58	4,128
qwen4b	25	338	3,370
qwen8b	33	347	807
SkillSeek (rerank)	288	**1,105**	**1,862**

Table 7. Per-call retrieval latency for every method we tested, sorted by median ascending. † qwen0_6b runs locally via sentence-transformers; the other Qwen variants are served through OpenRouter, which inflates p95.

Where the* 1.3 *s comes from. Table 7 decomposes the per-trial budget into per-call latency by retriever method. BM25 is essentially free (0.3 ms median); local bi-encoders sit in the tens of milliseconds; OpenRouter-served Qwen variants take hundreds. SkillSeek's cumulative trial overhead in Table 6 (1,342 ms) is dominated by a single skill_lookup call at 1,105 ms median, which is the cross-encoder rescoring 20 candidate pairs jointly. This is a real but bounded cost, comparable in absolute terms to OpenRouter-served retrievers but considerably more accurate (Table 5), and it is paid back in saved agent-loop time as already noted above.

Takeaway. Retrieval-equipped trials finish 39 s faster than no-skills runs despite 1.3 s of MCP overhead.

4 Related Work

Anthropic Agent Skills. The SKILL.md package format introduced by Anthropic in October 2025 packages procedural instructions, optional scripts, and reference files into a directory loaded progressively by the agent at runtime [15]. Li et al. [11] contribute SkillsBench, a benchmark of deterministically verifiable tasks paired with curated SKILL.md bundles, and report that curated skills add +16.2% on average across 7 model–harness configurations while 4+-skill bundles regress to +5.9%; we evaluate SkillSeek on the same testbed. Xu and Yan [15] survey the abstraction layer along four axes (architecture, acquisition, deployment, security) and propose a four-tier governance framework; Jiang et al. [9] formalize a skill as the four-tuple (C, π, T, R) generalizing the options framework and trace a seven-stage lifecycle. Both surveys explicitly distinguish SKILL.md packages from

RL options, code-only libraries, and RAG-distilled snippets, an abstraction we follow.

Empirical efficacy and skill selection. Beyond the curated SkillsBench setup, recent work shows that the headline gain is fragile. Han et al. [6] find that on 49 public software-engineering skills, 39 yield zero pass-rate change and the average gain is only +1.2%; the 7 skills that help carry domain-specific procedural knowledge absent from the model's prior. Concurrent with the present work, Liu et al. [13] stress-test agentic skill retrieval over a 34,000-skill pool harvested from public skill hubs, pairing `Qwen3-Embedding-4B` with BM25 via reciprocal rank fusion and delegating second-stage candidate selection to the agent itself; their study runs on Claude Code, Terminus-2, and Qwen-Code; we discuss how these compare to our OpenHands SDK choice in §2.3. They report that pass rate degrades monotonically as conditions become realistic and that only 49% of trajectories load all curated skills under defaults. SkillSeek occupies the complementary regime: a hand-curated 192-skill pool, a deterministic two-stage retriever (a bi-encoder followed by an off-the-shelf cross-encoder reranker, with no LLM in the retrieval loop), and a fully open-source, specification-faithful OpenHands harness, recovering within 1.7% of the per-task `oracle` bundle on SkillsBench at zero LLM-call overhead in the retrieval loop. We note that the `Qwen3-Embedding-4B` encoder underlying their hybrid is one of the bi-encoders our 10-task ablation (Table 5) shows our cross-encoder pipeline outperforms; the gap they report at higher recall comes from their LLM-mediated re-querying rather than from the encoder itself, and we trade that capability away for deployability.

Skill optimization, runtime, and security. A second line of work operates on the skills themselves rather than on selection. Gao et al. [4] compress descriptions and bodies via delta-debugging-based oracle preservation, achieving 48% description and 39% body compression with a +2.8% quality lift on a 600-skill evaluation. Gong et al. [5] cast skill-bundle tuning as bi-objective NSGA-II search over (pass rate, cost), achieving up to +131% pass rate and −31.7% cost on three SkillsBench SE tasks. Chen et al. [2] treat skills as code and the (model, harness) pair as a heterogeneous processor, applying capability-based compilation and JIT solidification for +15.3% task completion and up to −40% tokens. Shen et al. [14] build a closed-loop skill-mining system for scientific domains, validating 286 skills of which 71.1% are novel against existing libraries. On the security side, Liu et al. [12] introduce *guidance injection* attacks via `agent:bootstrap` hooks that achieve 16–64% success across six LLM backends with 94% evading existing scanners; Hou and Yang [8] propose a three-layer triage cascade reaching 0.800 F1 at $0.006/skill on a community marketplace; and Holzbauer et al. [7] use repository-context scoring to reduce single-skill scanner false positives, surfacing 121 skills pointing to abandoned, claimable repositories. SkillSeek is orthogonal to all of these: we select from a fixed pool at inference rather than mutating, optimizing, compiling, or scanning the skills, and the security thread above motivates the trust-aware extension we sketch in §5.

5 Conclusion

SkillSeek shows that off-the-shelf neural retrieval, exposed as an MCP server, recovers near-oracle quality on Anthropic-spec Agent Skills without any per-task human curation: in a fully open-source, specification-faithful stack on the 89-task SkillsBench, 0.467 task-mean pass rate, within 1.7% of the per-task hand-curated bundle and +8.3% over the no-skills baseline. The core architectural finding is that a 568 M-parameter cross-encoder reranker outperforms every larger bi-encoder we tested on this benchmark, and that the residual gap to oracle is a ranking failure on tasks whose pool contains multiple lexically and semantically similar near-twin skills, not a coverage failure. At deployment, SkillSeek adds about 1.3 s of CPU-only retrieval latency and +15% tokens per trial, runs at zero API cost, and does not slow the agent loop end-to-end.

Future work: trust-aware retrieval. SkillSeek today selects on relevance alone. The ecosystem this paper sits in is one in which marketplaces ship third-party skills at scale and recent attacks compromise the agent through them: Liu et al. [12] document guidance-injection attacks via `agent:bootstrap` hooks that succeed on 16–64% of trials across six LLM backends with 94% evading existing scanners; Holzbauer et al. [7] surface 121 marketplace-indexed skills pointing to abandoned, claimable GitHub repositories. Defensive infrastructure is starting to land: Hou and Yang [8] run a hierarchical triage cascade reaching 0.800 F1 at six tenths of a US cent per skill on a community marketplace. A natural next version of SkillSeek fuses retrieval with such a per-skill risk score: `skill_lookup` drops candidates whose pre-load audit exceeds a configurable threshold and emits an audit trail per result, so that the same plug-and-play interface that delivers near-oracle quality today can also enforce a trust policy as the ecosystem grows. We leave this trust-aware extension, and a paired evaluation against adversarial skill mixes, as future work.

References

[1] Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2025. M3-Embedding: Multi-Linguality, Multi-Functionality, Multi-Granularity Text Embeddings Through Self-Knowledge Distillation. arXiv:2402.03216 [cs] doi:10.48550/arXiv.2402.03216

[2] Le Chen, Erhu Feng, Yubin Xia, and Haibo Chen. 2026. SkVM: Revisiting Language VM for Skills across Heterogenous LLMs and Harnesses. arXiv:2604.03088 [cs] doi:10.48550/arXiv.2604.03088

[3] Zijian Chen, Xueguang Ma, Shengyao Zhuang, Ping Nie, Kai Zou, Andrew Liu, Joshua Green, Kshama Patel, Ruoxi Meng, Mingyi Su, Sahel Sharifymoghaddam, Yanxi Li, Haoran Hong, Xinyu Shi, Xuye Liu, Nandan Thakur, Crystina Zhang, Luyu Gao, Wenhu Chen, and

Jimmy Lin. 2025. BrowseComp-Plus: A More Fair and Transparent Evaluation Benchmark of Deep-Research Agent.
[4] Yudong Gao, Zongjie Li, Yuanyuanyuan, Zimo Ji, Pingchuan Ma, and Shuai Wang. 2026. SkillReducer: Optimizing LLM Agent Skills for Token Efficiency. arXiv:2603.29919 [cs] doi:10.48550/arXiv.2603.29919
[5] 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. arXiv:2604.09297 [cs] doi:10.48550/arXiv.2604.09297
[6] Tingxu Han, Yi Zhang, Wei Song, Chunrong Fang, Zhenyu Chen, Youcheng Sun, and Lijie Hu. 2026. SWE-Skills-Bench: Do Agent Skills Actually Help in Real-World Software Engineering? arXiv:2603.15401 [cs] doi:10.48550/arXiv.2603.15401
[7] Florian Holzbauer, David Schmidt, Gabriel Gegenhuber, Sebastian Schrittwieser, and Johanna Ullrich. 2026. Malicious Or Not: Adding Repository Context to Agent Skill Classification.
[8] Yinghan Hou and Zongyou Yang. 2026. SkillSieve: A Hierarchical Triage Framework for Detecting Malicious AI Agent Skills.
[9] Yanna Jiang, Delong Li, Haiyu Deng, Baihe Ma, Xu Wang, Qin Wang, and Guangsheng Yu. 2026. SoK: Agentic Skills – Beyond Tool Use in LLM Agents. arXiv:2602.20867 [cs] doi:10.48550/arXiv.2602.20867
[10] Chaofan Li, Zheng Liu, Shitao Xiao, and Yingxia Shao. 2023. Making Large Language Models A Better Foundation For Dense Retrieval. arXiv:2312.15503 [cs] doi:10.48550/arXiv.2312.15503
[11] 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. 2026. SkillsBench: Benchmarking How Well Agent Skills Work Across Diverse Tasks. arXiv:2602.12670 [cs] doi:10.48550/arXiv.2602.12670
[12] Fazhong Liu, Zhuoyan Chen, Tu Lan, Haozhen Tan, Zhenyu Xu, Xiang Li, Guoxing Chen, Yan Meng, and Haojin Zhu. 2026. Trojan's Whisper: Stealthy Manipulation of OpenClaw through Injected Bootstrapped Guidance. arXiv:2603.19974 [cs] doi:10.48550/arXiv.2603.19974
[13] Yujian Liu, Jiabao Ji, Li An, Tommi Jaakkola, Yang Zhang, and Shiyu Chang. 2026. How Well Do Agentic Skills Work in the Wild: Benchmarking LLM Skill Usage in Realistic Settings. arXiv:2604.04323 [cs] doi:10.48550/arXiv.2604.04323
[14] Shuaike Shen, Wenduo Cheng, Mingqian Ma, Alistair Turcan, Martin Jinye Zhang, and Jian Ma. 2026. SKILLFOUNDRY: Building Self-Evolving Agent Skill Libraries from Heterogeneous Scientific Resources. arXiv:2604.03964 [cs] doi:10.48550/arXiv.2604.03964
[15] Renjun Xu and Yang Yan. 2026. Agent Skills for Large Language Models: Architecture, Acquisition, Security, and the Path Forward. arXiv:2602.12430 [cs] doi:10.48550/arXiv.2602.12430
[16] Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025. Qwen3 Embedding: Advancing Text Embedding and Reranking Through Foundation Models. arXiv:2506.05176 [cs] doi:10.48550/arXiv.2506.05176

A Harness Adaptations

The setup paragraph in Section 3.1 states the strict experimental design (89 tasks, 5 conditions, 3 trials per condition). Several harness-implementation choices were necessary to make the sweep tractable; we record them here for completeness.

Agent backbone and serving stack. Generation used a local `gpt-oss-120b` served by vLLM on a single-host CPU/GPU rig, accessed by the agent through the OpenHands SDK direct-path with the `DockerDevWorkspace` backend. The agent-server, the model server, and the MCP skill server were three separate processes on the same host.

Per-trial caps. Each trial was capped at 30 agent turns and 600 s of wall-clock via `SIGALRM`; trials that hit either cap raise `TimeoutError` with a clean `result.json` so the verifier loop proceeds. Total wall clock for the 1,335-trial sweep at parallelism 3 was approximately 16 h through ∼406 M tokens, at $0 in API cost.

Docker overlay for non-root agent-server. SkillsBench task images install dependencies as `root` with `chmod700/root`, but `DockerDevWorkspace` runs the agent-server as a non-root user that needs read/write access under `/root`. We layer a one-line overlay built per-task on first encounter and cached thereafter:

```
FROM <task_image>:raw
RUN chmod -R o+rwX /root && \
    (chmod o+rx /root /home 2>/dev/null || true)
```

System-message suffix nudge. Without an explicit instruction to call `skill_lookup`, `gpt-oss-120b` ignores the MCP tools entirely. We append a suffix only when an MCP URL is configured (i.e., the `bm25/embed/rerank/qwen*` conditions); the `none/oracle/all` conditions do not receive it, keeping the A/B test clean.

150-***skill cap on*** **`all`.** The OpenHands SDK's WebSocket subscription times out at ∼173 skills loaded into the initial `AgentContext.skills`. Increasing `ws_timeout` from 30 to 300 seconds did not help, ruling out a time limit. Binary search localized the failure between 150 skills (works) and 173 skills (fails), confirming a frame-size ceiling. The `all` condition therefore uses the first 150 skills, in lexicographic order on skill name. MCP retrieval is unaffected because skills are streamed on demand rather than bulk-loaded.

Pass-rate parser. SkillsBench verifiers vary in pytest output format. Our `parse_reward()` attempts, in order: explicit `reward:X.XX`; `Npassed/Nfailed/Nerror` summary lines; `PASSED/FAILED/ERROR\b` per-test markers (verbose mode); pytest `[.FsExX]+` progress dots (dot mode). The `ERROR\b` branch was added mid-sweep after 15 trials were misclassified as `reward=None`.

B Implementation Details

1. 150-**vs**-192 **selection bias for the `all` baseline.** The WebSocket frame ceiling documented in Appendix A forces the `all` condition to load 150 of the 192 pool skills, selected in lexicographic order on skill name. This is *not* a uniform random sample: the alphabetical

Condition	Trials	Errors	ConvRun	Timeout	Runtime	Build
none	264	99	72	15	9	3
all	264	94	78	4	9	3
bm25	264	115	94	9	9	3
SkillSeek (rerank)	264	113	89	12	9	3
oracle	258	111	86	13	9	3

Table 8. Trial errors broken down by condition. The four error classes are agent-server-side ConversationRunError, 600 s per-trial TimeoutError, container-side RuntimeError ("Container stopped unexpectedly"), and BuildCommandError on the task Dockerfile build.

cutoff drops late-letter skills (e.g., xlsx-..., youtube-..., zoom-...), and skill-name spellings correlate weakly with task domain (e.g., several data-... skills cluster). The threat to validity is that all may slightly over- or under-represent particular domains relative to a uniform sample. Importantly, the headline comparison rerank vs. oracle does not depend on this baseline; the bound that *does* use it is the rerank > all comparison (Table 1), which is robust to the choice of 150 here because all mounts more skills than any reasonable retrieval set, distractor-dominated regardless of which 150 are chosen.

2. 233-**to**-192 **deduplication of the skill pool.** The 84 benchmark tasks ship 233 SKILL.md files in total, with a per-task bundle-size distribution from 1 to 7 (§3.2). Deduplication by skill name yields 192 unique skills; about 41 names are reused across multiple tasks. The 192-skill general pool indexed by SkillSeek is this deduplicated set.
3. **Some tasks have zero successful trials in any condition** (fix-druid-loophole-cve, energy-ac-optimal-power-flow). They run, the agent attempts, the agent-server errors. These are concentrated among heavyweight tasks and remain out of scope for this round.

C Error Patterns

Table 8 reports trial errors per condition. The total error rate hovers between 35 and 44% across all five conditions and is comparable for retrieval-equipped trials and the no-skills baseline, so retrieval does not destabilize the agent loop. The one notable shift is in TimeoutError, where all records only 4 timeouts versus none's 15; mounting many skills front-loads context and shortens trials, consistent with the wall-clock savings reported in Section 3.4. Container-level RuntimeError and BuildCommandError counts are condition-independent and concentrate on the same heavyweight tasks (fix-druid-loophole-cve: zero successes across 15 trials in any condition; energy-ac-optimal-power-flow: similar), out of scope for this work.