DREAMING THE RIGHT SKILL: CAPABILITY-GUIDED RETRIEVAL FOR LLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Retrieving reusable skills can improve LLM agents by providing procedural guidance for complex tasks. Existing retrieve-and-rerank pipelines match an agent's query directly to skill documents. As skill libraries grow, query-based matching can return many semantically related skills that provide no useful guidance for the current operation. We introduce DreamSkill, which uses a *Dream*—a short, query-conditioned description of the required capability, including its operations, intended outcomes, and explicit constraints—to guide both supervision construction and online retrieval. By decomposing compound requests into capability queries and matching each Dream to a supporting skill, we construct 260,631 step-level training pairs. A shared encoder learns to match Query+Dream to full skill documents and uses the same input format at deployment. During execution, the agent independently assesses each candidate against the task, observed state, and available tools before selecting among applicable procedures. One trained retriever serves different executors, each using its own base model for Dream generation and contextual selection. Across three benchmarks and three executors, DreamSkill achieves the highest mean task accuracy or success rate in all nine settings. Its retriever also leads all six usefulness metrics on skill pools of 6,660 and 35,059 documents. On Terminal-Bench, it improves Qwen3.5-9B accuracy by 5.99 percentage points over the strongest skill baselines. On Spider2-DBT, it raises DeepSeek-V4-Flash success from 44.79% without skills to 56.25%.

1 INTRODUCTION

LLM agents interact with external tools to manipulate files, execute commands, and complete multi-step tasks (Yao et al., 2023; Qin et al., 2024). Reusable skills package procedures, prerequisites, and supporting resources that let agents draw on existing solutions (Yuan et al., 2024; Wang et al., 2025). Well-matched skills can directly improve execution: on 20 Toolathlon-GYM tasks (Zhang et al., 2026), manually selected community skills raise a fixed executor's average score from 50.00% to 70.00% (Section 3.1). This motivates a practical question: how can an agent automatically acquire guidance that helps with its current operation?

Existing approaches retrieve and rerank skills by matching an agent's query to skill descriptions or full documents (Kang et al., 2026; Zheng et al., 2026; Wang et al., 2026b). As public skill collections reach tens of thousands of documents (Liu et al., 2026b), many candidates share topics and terminology while serving different operations. Query-based matching can return semantically related skills that provide no useful guidance for the current operation. In Figure 1(a), direct query-skill matching retrieves OAuth-related skills, but they do not provide Mermaid diagram-authoring guidance for the requested sequence diagram. Training poses a related problem: a compound task's labeled skills may support different operations, so each query needs its own supporting skill.

We introduce DreamSkill, which describes the capability sought as a short, query-conditioned *Dream* and uses it to guide both retrieval training and online search (Figure 2). The query specifies the concrete request; Dream describes the operations, intended outcomes, and explicit constraints that useful guidance should address. Offline, we decompose compound requests into queries for individual capabilities and match each Dream to a supporting skill in the original request's labeled set, producing 260,631 training pairs. The retriever learns to match Query+Dream to full skill documents and retains this input format at deployment. In Figure 1(b), Dream emphasizes the

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

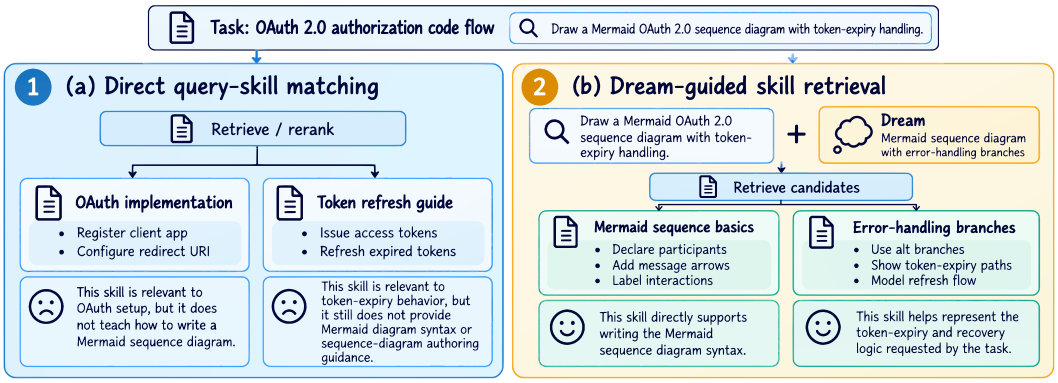

Figure 1: **Motivation for capability-guided skill retrieval.** *(a)* Direct query-skill matching retrieves semantically related OAuth skills, but they do not teach how to write the requested Mermaid sequence diagram. *(b)* Dream emphasizes the requested diagramming capability and error handling, guiding retrieval toward skills that directly support Mermaid sequence-diagram authoring and the required branches.

requested Mermaid diagramming and error handling, guiding retrieval toward skills that directly support sequence-diagram authoring. The executor then independently checks candidates against the task, observations, and tools before selecting applicable guidance. This accounts for whether a capability-matched procedure can be used under the current execution conditions. One trained retriever serves all evaluated executors; online Dream generation and selection use each executor's own base model without additional training.

We evaluate DreamSkill on SpreadsheetBench (Ma et al., 2024), Terminal-Bench (Merrill et al., 2026), and Spider2-DBT (Lei et al., 2025) with three executors searching the same 35,059-skill pool. It achieves the highest mean task accuracy or success rate in all nine settings, while its retriever leads all six usefulness metrics on two skill corpora. On Terminal-Bench with Qwen3.5-9B, accuracy reaches 28.46%, exceeding the strongest skill baselines by 5.99 percentage points. On Spider2-DBT, DreamSkill raises DeepSeek-V4-Flash success from 44.79% without skills to 56.25%.

2 Related Work

Skill retrieval. Agents reuse procedural knowledge through tool libraries and workflow memory (Yuan et al., 2024; Wang et al., 2024; 2025). SkillsBench tests whether curated guidance improves task execution (Li et al., 2026c), while Skill-Usage reports failures in retrieving and selecting usable guidance from roughly 34k web-collected skills (Liu et al., 2026b). SkillRet supplies task–skill training pairs (Kang et al., 2026); SkillRouter uses full skill bodies for retrieval and reranking (Zheng et al., 2026); R3-Skill learns query-conditioned compatibility from rejected skill combinations (Wang et al., 2026b). Our setting concerns retrieval from large skill libraries containing topically related candidates that may not support the current operation.

Generated retrieval inputs. HyDE embeds hypothetical documents (Gao et al., 2023), and Query2doc expands queries with generated pseudo-documents (Wang et al., 2023). Both use generated text to improve access to documents already present in a corpus. SkillReason transfers capability-reasoning supervision into a retriever that encodes the original query at deployment, without generating reasoning at search time (Jiang et al., 2026). SkillDreamer generates prospective skill descriptions, separately encodes them and the original query, and combines their retrieval scores with lexical refinement (Liu et al., 2026a). In DreamSkill, Dream helps match each decomposed training query to one supporting skill from its source task's labeled set. The retriever jointly encodes Query+Dream in both training and online search.

Selecting guidance for execution. SkillFlow judges guidance utility after retrieval and reranking and can reject candidates with mismatched tool requirements (Li et al., 2026a). SkillFeed conditions skill routing on task-relevant user profiles and document evidence (Wang et al., 2026a). In

DreamSkill, each candidate's full guidance and resources are assessed independently against the task, observations, query, and native tools before applicable candidates are compared. Useful partial guidance, such as a validation procedure, can qualify. The executor's own base model makes both decisions. We evaluate this acquisition process through both usefulness judgments and complete task execution.

3 METHOD

3.1 MOTIVATION

We first test whether task-matched procedural guidance improves a fixed executor. We evaluate Qwen3.5-9B on 20 warehouse analytics and reporting tasks from Toolathlon-GYM (Zhang et al., 2026). Matched Skill directly supplies manually selected, unmodified community skills; No Skill supplies no additional guidance. We run five attempts per task and condition, yielding 200 attempts in total. Appendix C.2 identifies the public skill sources.

Table 1: **Effect of task-matched skills (%).**

Guidance	Avg. score	Pass@3	Pass@5
No Skill	50.00	70.00	75.00
Matched Skill	**70.00**	**100.00**	**100.00**
Gain (pp)	**+20.00**	**+30.00**	**+25.00**

Table 1 shows that average score, measured over the 100 attempts per condition, rises from 50.00% to 70.00% with matched guidance, a gain of 20.00 percentage points (paired 95% CI: $[9.00, 30.00]$). Five-attempt task coverage (Pass@5) increases from 75.00% to 100.00%, corresponding to 15 versus all 20 tasks solved at least once. Matched guidance thus improves both average success across attempts and the range of tasks solved by the same executor.

To obtain such benefits automatically, an agent must identify guidance for its current operation within a large skill library. A topically related procedure can still be unhelpful, as illustrated in Figure 1. This leads to two design questions: how can the required capability guide both retrieval supervision and online search, and how can the agent select retrieved guidance that is applicable given its task, observations, and tools?

3.2 OVERVIEW AND PROBLEM SETUP

DreamSkill trains its retriever offline with Dream-guided pairs. Online, the executor generates Dreams for this fixed retriever and selects applicable guidance (Figure 2).

Let x be the task, o_t the observations available at search step t, and $\mathcal{T}$ the executor's native tools. The agent issues a query q_t describing the requested deliverable, observed facts, and the guidance sought. A step-level need is a retrievable capability and may span several tool actions. Each skill s in corpus $\mathcal{S}$ contains a name, description, original body, and optional resources. A call to `search_skill` activates one skill or returns NONE. The task can use multiple skills through successive calls, and adopted references remain available. Offline request decomposition constructs training queries; online searches arise from the executor's current work.

A Dream $d = D(q)$ describes the capability that useful guidance should provide. It is generated solely from q and expresses the relevant operations, objects, outcomes, and explicit constraints. The query states the request, the Dream describes the procedural help sought, and the retrieved skill supplies the instructions for carrying it out.

3.3 STEP-LEVEL SUPERVISION CONSTRUCTION

We assign a training positive to each capability need. A compound request can be associated with several gold skills, each supporting a different operation. Matching its derived needs separately makes the training target specific to the procedure required by each query.

For a SkillRet request u, let $G(u)$ contain its original skills labeled relevant (Kang et al., 2026). A generator produces standalone capability queries $\{q_j\}_{j=1}^{m(u)}$. Each query retains the tools, objects, and constraints needed to understand its operation. Shared context is repeated where necessary, and operations that serve one capability remain together. The generator then produces a Dream $d_j =$

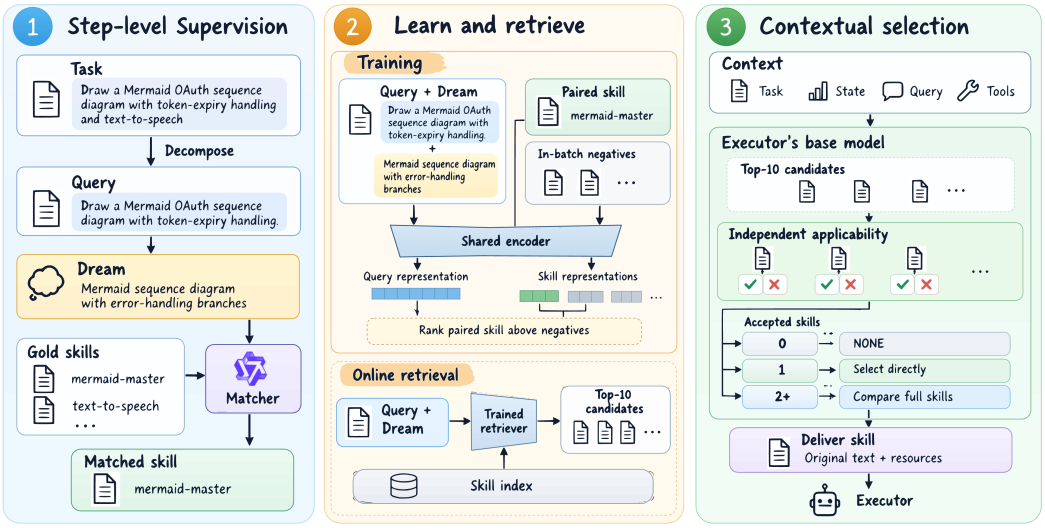

Figure 2: **Overview of DreamSkill.** *Left:* Dream-guided matching selects one supporting source gold skill for each query. *Middle:* A shared encoder learns to match Query+Dream to full skill documents and retrieves Top-10 candidates. *Right:* The executor checks candidates independently in execution context, then returns NONE, adopts the sole applicable skill, or compares multiple applicable skills. Selected original text and resources are delivered for execution.

$D(q_j)$ for each query. Decomposition and Dream generation use the request; the gold documents enter at matching.

The matcher receives Dream d_j as its request-side input and the full gold documents in $G(u)$. It selects the document that best matches the described capability, technical objects or APIs, decisive operations, and intended outcome. Different wording and more general reusable implementations qualify when they directly support that capability, rather than merely sharing its topic. When several candidates match, the model chooses the best overall match across these criteria. When none directly matches, it returns NONE. The retained training set is

$$\mathcal{D} = \{(q_j, d_j, s_j^+) : \; s_j^+ = R(d_j, G(u)) \neq \text{NONE}\}. \tag{1}$$

Each retained need contributes one pair, with a single supporting skill as its positive.

The supervision panel of Figure 2 condenses a real training example. The source request asks for a Mermaid sequence diagram of an OAuth 2.0 authorization-code flow with token-expiration handling, followed by audio of its title and node descriptions. Its gold set contains `mermaid-master` and `text-to-speech`. For the diagram-generation query, the matcher selects `mermaid-master`: its instructions on participants, messages, and control flow, including `alt`, support the requested diagram. The speech skill serves the other operation. This positive pairs a specific diagram request with a reusable construction procedure. Appendix A.1 provides the query, Dream, and matched excerpt.

Offline decomposition, Dream generation, and matching use Qwen3-32B (Qwen Team, 2025). The procedure yields 260,631 equally weighted training pairs. Appendix A gives the matching instruction, example, and skill sources.

3.4 Learning and Retrieving with Dream

We train the retriever to match the request and its Dream to a full skill document. The request preserves the query's objects and constraints, while Dream makes the desired procedural capability explicit. A shared Qwen3-Embedding-8B encoder f_θ (Zhang et al., 2025) learns this match. Let $F_Q(q, d)$ denote the joint Query+Dream input and $F_S(s)$ the skill's name, description, and original body. We compute

$$\mathbf{z}_{q,d} = \text{norm}\big(f_\theta(F_Q(q, d))\big), \qquad \mathbf{z}_s = \text{norm}\big(f_\theta(F_S(s))\big), \tag{2}$$

where $\mathrm{norm}(\mathbf{v}) = \mathbf{v}/\|\mathbf{v}\|_2$. Query and Dream form one text input. Skill documents are encoded separately with the same weights, so the learned match can depend on their operations, prerequisites, and checks.

Training. For a batch of B matched pairs, we use a multiple-negatives ranking objective:

$$\mathcal{L}_{\mathrm{ret}} = -\frac{1}{B}\sum_{i=1}^{B} \log \frac{\exp(\mathbf{z}_{q_i,d_i}^{\top}\mathbf{z}_{s_i^+}/\tau)}{\sum_{j=1}^{B}\exp(\mathbf{z}_{q_i,d_i}^{\top}\mathbf{z}_{s_j^+}/\tau)}. \tag{3}$$

Here τ is the temperature. The objective ranks the skill assigned to the current need above other paired documents in the batch, following contrastive dense retrieval (Karpukhin et al., 2020). LoRA (Hu et al., 2022) updates the shared encoder, producing DreamSkill-Embedding-8B. The executor models retain their original parameters. Appendix A.3 summarizes retriever training.

Online retrieval. At a search call, the executor's base model generates $d_t = D_{\mathrm{exec}}(q_t)$. The trained encoder uses the same Query+Dream format as in training and retrieves the Top-10 skills from a FAISS index (Johnson et al., 2021) of normalized skill embeddings:

$$C_t = \mathrm{Top10}_{s\in\mathcal{S}}\ \mathbf{z}_{q_t,d_t}^{\top}\mathbf{z}_s. \tag{4}$$

The shared input format lets different executors express their current needs to one trained retriever through their own Dream generators. The Top-10 original documents and resource lists pass to contextual selection. Appendix E provides the Dream prompt and retrieval input template.

3.5 Selecting a skill in execution context

Retrieval ranks procedures by their match to the requested capability. Adoption also depends on where and how the agent will use them. Desktop and headless recalculation, for example, serve the same operation through different interfaces. We first establish whether each procedure is applicable, then choose among the applicable procedures.

For each $s \in C_t$, the executor's base model reads the full document and resource list together with $(x, o_t, q_t, \mathcal{T})$. It predicts a binary label $a_t(s)$ and identifies supporting sections. A candidate is applicable when it supplies procedural help under the observed conditions. Useful partial guidance, such as a validation method, can qualify. The selector reads only existing observations; it neither issues internal terminal checks nor pre-installs candidate skills. The accepted set is

$$A_t = \{s \in C_t : a_t(s) = 1\}. \tag{5}$$

Assessing candidates independently establishes their applicability before they are compared for preference. The number accepted determines the three branches in Figure 2:

$$s_t = \begin{cases} \texttt{NONE}, & |A_t| = 0, \\ \text{the sole accepted skill}, & |A_t| = 1, \\ \mathrm{Select}_{\mathrm{exec}}(x, o_t, q_t, \mathcal{T}, A_t), & |A_t| > 1 \end{cases}. \tag{6}$$

When several candidates pass, the executor compares their full original guidance and resources in the same execution context. It selects the reference that best supports the task and observed state, returning one identifier from A_t. The supporting sections ground applicability judgments; the final comparison reads the accepted documents in full. Appendix B gives the selection instruction.

Selected guidance and resource entry points are added to subsequent system messages. The skill bundle is copied to the sandbox for use through the executor's existing terminal. Short documents are supplied in full; longer ones use original-section selection (Appendix B). Later searches may add skills; NONE preserves prior activations. Dream generation and contextual decisions use the executor's own base model.

4 Experiments

We evaluate whether Dream-guided skill acquisition improves task accuracy across models and tasks. Retrieval measurements and component comparisons examine the roles of positive assignment, learned representations, and contextual selection.

4.1 EXPERIMENTAL SETUP

Tasks and executors. We evaluate Qwen3.5-9B and Qwen3.5-27B (Qwen Team, 2026), together with DeepSeek-V4-Flash (DeepSeek-AI, 2026), on SpreadsheetBench Verified (400 tasks) (Ma et al., 2024), Terminal-Bench 2.1 (89 tasks) (Merrill et al., 2026), and Spider2-DBT (64 tasks) (Lei et al., 2025), covering spreadsheet editing, terminal work, and database transformations. Skill-enabled methods use the same 35,059-skill pool. We run three attempts per task: Accuracy/SR measures successful episodes, while supplementary Pass@3 measures tasks solved at least once. DreamSkill reuses one trained retriever across executors. Appendix C.1 gives the sampling settings.

Skill acquisition. Skill-enabled methods obtain their first `search_skill` result before normal execution, optionally after one read-only terminal observation, and may search again later. No Skill bypasses retrieval.

Baselines. Task comparisons include No Skill, BM25, Qwen3 retrieval, SkillRet, R3-Skill, SkillRouter, SkillReason, and SkillFlow (Robertson & Zaragoza, 2009; Zhang et al., 2025; Kang et al., 2026; Wang et al., 2026b; Zheng et al., 2026; Jiang et al., 2026; Li et al., 2026a). We follow each baseline's original input construction, retrieval, reranking, and selection procedures. All methods share an OpenEnv-based agent loop (Spisak et al., 2025), benchmark-specific executor prompts, and native verifiers; LLM-based query generation and selection use the corresponding executor's base model where required. Appendix C.3 gives model identities and retrieval and delivery counts.

Table 2: **Task execution across three executors and three benchmarks.** DreamSkill achieves the highest mean Accuracy/SR in all nine settings. Panel (b) reports supplementary Pass@3 results.

	Qwen3.5-9B			Qwen3.5-27B			DeepSeek-V4-Flash		
(a) Accuracy / SR (%)									
Method	SS	TB	DBT	SS	TB	DBT	SS	TB	DBT
No Skill	34.25	21.35	13.54	63.83	35.58	30.73	82.42	63.30	44.79
BM25	33.83	22.47	11.98	63.92	28.46	28.65	82.33	61.80	52.08
Qwen3 retrieval	34.67	20.22	13.54	65.25	30.34	28.65	83.67	60.67	53.65
SkillRet	35.25	22.47	15.63	64.17	32.58	30.73	82.92	64.04	54.17
R3-Skill	35.33	20.97	14.58	64.67	28.46	28.65	83.92	64.04	50.52
SkillRouter	37.92	20.97	12.50	65.33	31.46	26.56	83.83	62.55	52.60
SkillReason	34.08	20.97	13.02	65.33	35.21	29.69	82.67	63.30	54.17
SkillFlow	33.83	16.85	14.58	66.92	35.96	25.00	83.00	62.55	54.17
DreamSkill	**41.58**	**28.46**	**18.75**	**68.42**	**37.45**	**31.25**	**88.33**	**68.16**	**56.25**
(b) Pass@3 (%)									
Method	SS	TB	DBT	SS	TB	DBT	SS	TB	DBT
No Skill	54.00	32.58	25.00	78.50	**46.07**	**39.06**	90.25	77.53	51.56
BM25	52.75	29.21	20.31	78.75	43.82	**39.06**	91.00	71.91	59.38
Qwen3 retrieval	54.00	32.58	25.00	80.00	43.82	**39.06**	92.50	75.28	**60.94**
SkillRet	54.25	33.71	21.88	77.75	**46.07**	37.50	91.50	76.40	**60.94**
R3-Skill	54.75	29.21	23.44	79.25	**46.07**	35.94	90.75	77.53	57.81
SkillRouter	52.25	26.97	21.88	**80.25**	**46.07**	35.94	91.25	75.28	**60.94**
SkillReason	54.00	31.46	25.00	78.25	**46.07**	**39.06**	91.25	77.53	**60.94**
SkillFlow	52.25	28.09	23.44	80.00	**46.07**	37.50	90.50	75.28	59.38
DreamSkill	**59.25**	**40.45**	**32.81**	**80.25**	**46.07**	**39.06**	**93.50**	**78.65**	**60.94**

Notes. SS = SpreadsheetBench Verified; TB = Terminal-Bench 2.1; DBT = Spider2-DBT. Panel (a) reports SS/TB Accuracy and DBT success rate. All values are percentages. Three attempts are used per task. Bold denotes the best score within each model and benchmark, including ties.

4.2 TASK EXECUTION RESULTS

DreamSkill achieves the highest mean task Accuracy/SR in all nine benchmark–executor settings (Table 2). With Qwen3.5-9B, it reaches 41.58% on SpreadsheetBench, 28.46% on Terminal-Bench, and 18.75% on Spider2-DBT. The paired gains over the strongest skill baselines are 3.67, 5.99,

and 3.13 percentage points, respectively. The complete acquisition method therefore improves task completion across spreadsheet editing, terminal work, and database transformations.

The same trained retriever supports gains with larger executors. On Terminal-Bench, Qwen3.5-27B reaches 37.45%, compared with 35.96% for the strongest skill baseline. With DeepSeek-V4-Flash, Spider2-DBT success rises from 44.79% without skills to 56.25%, exceeding the strongest skill baseline at 54.17%. Tables 12 and 13 report trial variability and paired intervals against No Skill and the strongest skill baselines.

4.3 Skill retrieval results

Retrieval evaluation. We evaluate 6,460 held-out capability queries reconstructed from the Skill-Ret test split (Kang et al., 2026) on pools of 6,660 and 35,059 skills. For this static evaluation, Qwen3.5-9B (Qwen Team, 2026) generates cached Dreams from the queries with thinking disabled, without access to gold or candidate skills. Separately, Qwen3-32B judges usefulness from the query and candidate document, without Dream, gold labels, or retrieval scores. Useful skills provide concrete procedural help, including alternative methods and partial guidance. JU@1/5/10 measures access to a useful skill; MRR@10 averages the reciprocal rank of the first useful skill; P@5/10 measures the useful fraction of the list. Unlike these query-level judgments, online applicability also uses execution context. Appendix C.5 details the fixed Dream inputs and metrics. The judge achieves 88.51% F1 against adjudicated human labels (Appendix C.5).

Table 3: **Useful-skill retrieval on two skill pools.** Embedding models use no reranker; BM25 is lexical. JU@k and P@k are percentages; MRR@10 is on $[0, 1]$. Bold marks the best.

A. Pool size: 6,660 skills						
Retriever	JU@1	JU@5	JU@10	MRR@10	P@5	P@10
BM25	67.80	92.29	95.73	.7818	52.88	45.98
Qwen3-Embedding-8B	75.54	95.53	97.59	.8426	62.18	55.97
SKILLRET-Embedding-8B	76.21	96.11	98.27	.8488	62.06	55.68
R3-embedding-0.6b	77.17	96.15	98.20	.8549	63.11	56.76
SkillRouter-Embedding-0.6B	67.72	92.11	95.99	.7823	53.49	47.07
SkillReason-embedding-4b	77.93	96.13	98.07	.8593	61.96	55.22
bge-base-en-v1.5	62.54	88.93	93.79	.7379	47.50	41.06
DreamSkill-Embedding-8B	**81.80**	**97.54**	**98.81**	**.8886**	**69.03**	**62.97**
B. Pool size: 35,059 skills						
Retriever	JU@1	JU@5	JU@10	MRR@10	P@5	P@10
BM25	65.06	90.88	95.09	.7613	58.17	54.19
Qwen3-Embedding-8B	73.59	94.43	96.92	.8267	66.95	63.11
SKILLRET-Embedding-8B	65.71	93.75	97.03	.7778	62.14	59.20
R3-embedding-0.6b	71.67	95.26	97.91	.8193	65.97	62.27
SkillRouter-Embedding-0.6B	60.62	89.46	94.63	.7281	53.95	50.74
SkillReason-embedding-4b	72.76	95.57	97.99	.8273	66.53	62.51
bge-base-en-v1.5	60.39	87.54	93.20	.7200	51.86	47.93
DreamSkill-Embedding-8B	**77.35**	**97.11**	**98.58**	**.8605**	**71.99**	**68.83**

JU measures access to at least one useful skill; P measures the useful fraction of a candidate list. bge-base-en-v1.5 (Xiao et al., 2024) is the pretrained encoder used by SkillFlow (Li et al., 2026a). Annotation details appear in Appendix C.5.

DreamSkill-Embedding-8B leads all six usefulness metrics on both pools (Table 3). Its first-ranked skill is useful for 81.80% and 77.35% of queries with the 6,660- and 35,059-skill pools, respectively, compared with 77.93% and 73.59% for the strongest respective baselines. The learned representation thus places a useful procedure at the first rank more often.

The improvement also extends to the candidate list. P@10 reaches 62.97% and 68.83% with the 6,660- and 35,059-skill pools, exceeding the strongest baselines by 6.21 and 5.72 percentage points. JU@5 reaches 97.54% and 97.11%, respectively. These results address two aspects of useful-skill retrieval in a large library: finding a useful procedure near the top and increasing the proportion of useful guidance available to subsequent selection.

4.4 ABLATION STUDY

We compare the complete method C with variants of step-level matching (Table 4), retrieval learning and representation (Table 5), and contextual selection (Table 6). Task ablations use Qwen3.5-9B and the full-model reference from Table 2. Retrieval ablations report JU@1, JU@5, and MRR@10 on the 35,059-skill pool. All task variants run complete episodes from the same initial tasks. Appendix D provides complete results, paired confidence intervals, and the shared comparison conditions.

Table 4: Step-level matching with Query+Dream, full-document encoding, and the training objective held fixed. U uses a uniformly sampled source-task gold skill as its positive. Task scores use Qwen3.5-9B; values except MRR@10 are percentages.

	Task execution			Retrieval		
Embedding model / variant	SS Acc.	TB Acc.	DBT SR	JU@1	JU@5	MRR@10
DreamSkill-Embedding-8B (C)	**41.58**	**28.46**	**18.75**	**77.35**	**97.11**	**.8605**
DreamSkill-Embedding-8B (U)	38.42	27.34	17.19	70.68	95.50	.8143

U: w/o step-level matching. Complete retrieval and task results appear in Tables 9 and 11.

Step-level matching. Dream-guided positive assignment improves both retrieval usefulness and task accuracy. U retains C's capability queries, Dreams, training configuration, and single-positive objective, but samples one skill uniformly from the source task's gold set for each record. The sampled pairs are fixed before training. Table 4 compares this task-level assignment with the full model's matched positives.

Matching the positive to the current need yields a paired JU@1 gain of 6.67 percentage points on the 35,059-skill pool (95% CI: $[5.67, 7.71]$). Task Accuracy/SR improves on all three benchmarks, including 3.17 points on SpreadsheetBench. All six retrieval metrics improve on both pools (Table 9). With the queries and Dreams held fixed, the comparison supports learning which source procedure addresses the individual need, rather than choosing a positive solely from task-level relevance labels.

Table 5: Retriever learning and representation with the full contextual selector and Qwen3.5-9B. Task Accuracy/SR and JU values are percentages; MRR@10 is on $[0, 1]$.

	Task execution			Retrieval		
Embedding model / variant	SS Acc.	TB Acc.	DBT SR	JU@1	JU@5	MRR@10
Qwen3-Embedding-8B (F)	39.83	28.09	14.58	76.18	95.68	.8477
DreamSkill-Embedding-8B (B)	37.00	**28.46**	13.02	76.95	96.84	.8573
DreamSkill-Embedding-8B (D)	38.50	28.09	13.02	73.59	95.42	.8318
DreamSkill-Embedding-8B (C)	**41.58**	**28.46**	**18.75**	**77.35**	**97.11**	**.8605**

F: original checkpoint + Dream; B: Query-only; D: metadata embedding; C: full Query+Dream / full-document model. B/C/D share the same matched pairs; C repeats the full model in Table 2. SS/TB/DBT follow Table 2; complete metrics appear in Appendix D.

Retriever learning and representation. Learning the retriever improves task accuracy with the Dream-generation and contextual-selection policies held fixed. F uses the original Qwen3-Embedding-8B checkpoint with Query+Dream and full documents. Compared with F, C raises JU@5 from 95.68% to 97.11% and P@10 from 65.38% to 68.83% on the 35,059-skill pool. All six retrieval metrics favor C on both pools (Table 9). The learned model also raises SpreadsheetBench Accuracy from 39.83% to 41.58% and Spider2-DBT SR from 14.58% to 18.75% under the same full selector (Table 5).

Dream in the retrieval input. With matched training positives held fixed, retaining Dream at training and retrieval increases SpreadsheetBench Accuracy by 4.58 percentage points and Spider2-DBT SR by 5.73 points over Query-only input (B). This isolates the contribution of encoding Dream from

the choice of supporting skills. JU@1, JU@5, and MRR@10 also increase; paired task intervals appear in Appendix D.

C/U and C/B separate supervision from representation: C/U changes the supporting positive while preserving queries and Dreams; C/B changes whether Dream is encoded while preserving matched positives. The task gains support need-specific supervision and retaining the capability description during retrieval.

Skill document representation. Full-document encoding improves SpreadsheetBench Accuracy by 3.08 points and Spider2-DBT SR by 5.73 points over metadata encoding (D). Both models receive Query+Dream, but D represents each skill through its name and description. Encoding the body raises JU@1 from 73.59% to 77.35%.

Table 6: Task execution with the retriever fixed and the selection policy varied. All conditions use DreamSkill-Embedding-8B and Qwen3.5-9B. Values are percentages.

	SpreadsheetBench		Terminal-Bench		Spider2-DBT	
Selection policy	Acc.	Pass@3	Acc.	Pass@3	SR	Pass@3
Embedding Top-1	35.58	55.00	24.34	35.96	15.10	23.44
Joint selection + NONE	40.33	57.75	27.34	**41.57**	15.10	26.56
Metadata-only selection	36.50	55.25	23.22	34.83	14.06	23.44
Forced selection	38.50	57.75	26.97	40.45	16.67	26.56
Full contextual selection	**41.58**	**59.25**	**28.46**	40.45	**18.75**	**32.81**

The full row is the common C reference for the task ablations. Bold marks the best value in each column, including ties. Policy definitions appear in Appendix D.2.

Contextual selection. Selecting guidance in execution context outperforms direct rank-one adoption on all three benchmarks (Table 6). Both policies use the same trained retriever. The full policy improves Accuracy/SR over Embedding Top-1 by 6.00, 4.12, and 3.65 percentage points on SpreadsheetBench, Terminal-Bench, and Spider2-DBT.

Separating applicability from relative preference also yields higher mean task accuracy than Joint selection + NONE on all three benchmarks. Joint selection reads the Top-10 full documents and execution context in one call and selects a candidate or NONE. The full policy first judges each candidate independently and compares the accepted documents when several pass. On Spider2-DBT, this policy raises SR from 15.10% to 18.75%.

The full policy also outperforms metadata-only and forced selection in task Accuracy/SR. Metadata-only selection bases adoption on names and descriptions, while forced selection requires a reference to be adopted. On Spider2-DBT, full-document contextual selection reaches 18.75%, compared with 14.06% and 16.67%, respectively.

Document encoding and document inspection contribute at different stages. C/D changes the information available to the retriever while retaining the full selector; metadata-only selection changes the information used for adoption while retaining the full retriever. Their task results support representing procedural content when retrieving candidates and inspecting that content when deciding which guidance to use.

5 Conclusion

DreamSkill retrieves useful procedural guidance from large skill libraries where semantically related skills may not support the current operation. Dream specifies the required capability, guides step-level positive assignment, and is jointly encoded with the query during retriever training and online search. The executor independently assesses applicability in execution context before selecting original guidance. With one trained retriever shared across executors, DreamSkill achieves the highest mean task accuracy or success rate in all nine benchmark–executor settings using a 35,059-skill pool. Its retriever also leads all six usefulness metrics on both evaluated pools. These results show how capability-guided retrieval learning and contextual selection can turn large skill libraries into effective guidance for task execution.

REPRODUCIBILITY STATEMENT

An anonymized version of our source code is included in the supplementary material. Appendices A–C specify data construction, retriever training, model roles, skill selection, benchmark scoring, and metrics. Appendix D defines the component comparisons, including the step-level matching ablation, and reports complete metrics and uncertainty statistics. Appendix E provides the search instructions, Dream-generation prompt, and input formats.

AI USE STATEMENT

AI tools assisted with data annotation, identifying related work, manuscript polishing, and code implementation.

REFERENCES

Anthropic. Knowledge Work Plugins. Software repository, n.d. URL https://github.com/anthropics/knowledge-work-plugins.

Daymade. Claude Code Skills Marketplace. Software repository, n.d. URL https://github.com/daymade/claude-code-skills.

DeepSeek-AI. DeepSeek-V4: Towards highly efficient million-token context intelligence, 2026. URL https://arxiv.org/abs/2606.19348.

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. Precise zero-shot dense retrieval without relevance labels. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1762–1777, 2023. doi: 10.18653/v1/2023.acl-long.99. URL https://aclanthology.org/2023.acl-long.99/.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Jeffallan. Claude Skills. Software repository, n.d. URL https://github.com/Jeffallan/claude-skills.

Donghong Jiang, Endian Lin, Luoping Cui, Hanqing Liu, Mingjie Liu, Fan Yang, Hong Wang, Zhao Yang, and Chuang Zhu. SkillReason: Reasoning-enhanced agent skill retrieval for implicit user requests. *arXiv preprint arXiv:2608.08640*, 2026. doi: 10.48550/arXiv.2608.08640. URL https://arxiv.org/abs/2608.08640v1.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2021. doi: 10.1109/TBDATA.2019.2921572. URL https://doi.org/10.1109/TBDATA.2019.2921572.

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

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 6769–6781, 2020. doi: 10.18653/v1/2020.emnlp-main.550. URL https://aclanthology.org/2020.emnlp-main.550/.

Fangyu Lei, Jixuan Chen, Yuxiao Ye, Ruisheng Cao, Dongchan Shin, Hongjin Su, Zhaoqing Suo, Hongcheng Gao, Wenjing Hu, Pengcheng Yin, Victor Zhong, Caiming Xiong, Ruoxi Sun, Qian Liu, Sida Wang, and Tao Yu. Spider 2.0: Evaluating language models on real-world enterprise text-to-SQL workflows. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/46c10f6c8ea5aa6f267bcdabcb123f97-Abstract-Conference.html.

Fangzhou Li, Pagkratios Tagkopoulos, and Ilias Tagkopoulos. SkillFlow: Scalable and efficient agent skill retrieval system. In *Third Conference on Language Modeling*, 2026a. URL https://openreview.net/forum?id=79D1g9mNrm.

Junlong Li, Wenshuo Zhao, Jian Zhao, Weihao Zeng, Haoze Wu, Xiaochen Wang, Rui Ge, Yuxuan Cao, Yuzhen Huang, Wei Liu, Junteng Liu, Zhaochen Su, Yiyang Guo, Fan Zhou, Lueyang Zhang, Juan Michelini, Xingyao Wang, Xiang Yue, Shuyan Zhou, Graham Neubig, and Junxian He. The tool decathlon: Benchmarking language agents for diverse, realistic, and long-horizon task execution. In *International Conference on Learning Representations*, 2026b. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/5783212d85c205ef823b8974d44872c5-Abstract-Conference.html.

Xiangyi Li, Yimin Liu, Wenbo Chen, et al. SkillsBench: Benchmarking how well agent skills work across diverse tasks. *arXiv preprint arXiv:2602.12670*, 2026c. URL https://arxiv.org/abs/2602.12670v4. Version 4.

Shuo Liu, Yutong Yang, Haohao Xiao, Mouxing Yang, and Xi Peng. Imagine before retrieval: Prospective skill retrieval for LLM agents. *arXiv preprint arXiv:2609.01642*, 2026a. URL https://arxiv.org/abs/2609.01642v1.

Yujian Liu, Jiabao Ji, Li An, Tommi Jaakkola, Yang Zhang, and Shiyu Chang. How well do agentic skills work in the wild: Benchmarking LLM skill usage in realistic settings. In *Third Conference on Language Modeling*, 2026b. URL https://arxiv.org/abs/2604.04323.

Zeyao Ma, Bohan Zhang, Jing Zhang, Jifan Yu, Xiaokang Zhang, Xiaohan Zhang, Sijia Luo, Xi Wang, and Jie Tang. SpreadsheetBench: Towards challenging real world spreadsheet manipulation. In *Advances in Neural Information Processing Systems*, volume 37, pp. 94871–94908, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/ac840df270ac537dd74530a15c332684-Abstract-Datasets_and_Benchmarks_Track.html.

Mike Merrill, Alexander Shaw, Nicholas Carlini, et al. Terminal-Bench: Benchmarking agents on hard, realistic tasks in command line interfaces. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/444a3737adaee10d86ad2ef5f74468e6-Abstract-Conference.html.

Nous Research. Hermes Agent. Software repository, n.d. URL https://github.com/NousResearch/hermes-agent.

OpenAI. Plugins. Software repository, n.d. URL https://github.com/openai/plugins.

OpenClaw contributors. OpenClaw. Software repository, n.d. URL https://github.com/openclaw/openclaw.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/28e50ee5b72e90b50e7196fde8ea260e-Abstract-Conference.html.

Qwen Team. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

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

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 3982–3992, 2019. doi: 10.18653/v1/D19-1410. URL https://aclanthology.org/D19-1410/.

Stephen E. Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009. doi: 10.1561/1500000019. URL https://dblp.org/rec/journals/ftir/RobertsonZ09.

Joseph Spisak, Davide Testuggine, Zach Wentz, Pierre Andrews, Sanyam Bhutani, Hamid Shojanazeri, Pankit Thapar, Emre Guven, Lewis Tunstall, and Vaibhav Srivastav. Building the open agent ecosystem together: Introducing OpenEnv. Hugging Face Blog, 2025. URL https://huggingface.co/blog/openenv.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=ehfRiF0R3a.

Liang Wang, Nan Yang, and Furu Wei. Query2doc: Query expansion with large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 9414–9423, 2023. doi: 10.18653/v1/2023.emnlp-main.585. URL https://aclanthology.org/2023.emnlp-main.585/.

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

Zifei Wang, Wei Wen, Qiang Ji, Keyu Chen, Ruizhi Qiao, and Xing Sun. Skill is not document: Query-conditioned compatibility for LLM agent skill routing. *arXiv preprint arXiv:2606.03565*, 2026b. URL https://arxiv.org/abs/2606.03565v5. Version 5.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 63897–63911, 2025. URL https://proceedings.mlr.press/v267/wang25bx.html.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. C-Pack: Packed resources for general Chinese embeddings. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 641–649, 2024. doi: 10.1145/3626772.3657878. URL https://doi.org/10.1145/3626772.3657878.

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

Lifan Yuan, Yangyi Chen, Xingyao Wang, Yi Fung, Hao Peng, and Heng Ji. CRAFT: Customizing LLMs by creating and retrieving from specialized toolsets. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/af31604708f3e44b4de9fdfa6dcaa9d1-Abstract-Conference.html.

Puzhen Zhang, Weijie Bai, Wendong Fan, and Guohao Li. Toolathlon-GYM: Large-scale long-horizon environments for tool-use agents, 2026. URL https://github.com/eigent-ai/toolathlon_gym.

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. Qwen3 Embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025. doi: 10.48550/arXiv.2506.05176. URL https://arxiv.org/abs/2506.05176.

Yanzhao Zheng, Zhentao Zhang, Chao Ma, Yuanqiang Yu, Jihuai Zhu, Yong Wu, Tianze Xu, Baohua Dong, Hangcheng Zhu, Ruohui Huang, and Gang Yu. SkillRouter: Skill routing for LLM agents at scale. In *Third Conference on Language Modeling*, 2026. URL https://arxiv.org/abs/2603.22455.

ZJUNLP. Skills. Software repository, n.d. URL https://github.com/ZJUNLP/Skills.

A Supervision and Skill Resources

A.1 Dream-guided supervision

Training and evaluation requests follow their respective SkillRet source splits (Kang et al., 2026). Each retained training record pairs a capability query and its Dream with one supporting gold skill, as defined in Section 3.3. A Dream expresses the required operations, objects, outcomes, and explicit constraints. Qwen3-32B generates the offline queries and Dreams and matches each Dream to a gold document. Its core matching instruction is:

> Select the one candidate gold skill document that best matches the hypothetical skill excerpt.
>
> Match the capability, technical objects or APIs, decisive operations, and intended outcome. Different wording or a more general reusable implementation is acceptable; merely sharing a topic is not.

The matcher compares the Dream with the candidate gold documents in full and selects the best overall match. If none directly matches, it returns NONE and the record is excluded from training.

The supervision panel of Figure 2 summarizes a real training example whose original request is:

> Can you generate a Mermaid sequence diagram showing the OAuth 2.0 authorization code flow between a mobile client, an API gateway, and an identity provider, including error handling for token expiration, and then convert the diagram's title and node descriptions into an audio file so I can listen to the flow while commuting?

The gold set contains `mermaid-master` and `text-to-speech`. For diagram generation, the derived input is:

> **Query:** Generate a Mermaid sequence diagram for OAuth 2.0 authorization code flow with mobile client, API gateway, identity provider, and error handling for token expiration.
>
> **Dream:** Generate a Mermaid sequence diagram illustrating the OAuth 2.0 authorization code flow involving a mobile client, API gateway, and identity provider, including explicit error handling for token expiration scenarios.

The positive is `mermaid-master`. Its *Sequence Diagrams* section provides participant ordering, message syntax, and control-flow constructs such as `alt`. The speech skill supports the source request's separate audio operation. The assignment therefore connects a specific execution need to a reusable procedure, rather than inheriting all of the source task's relevant skills.

A.2 Skill pools

We use the 6,660-skill candidate pool from SkillRet's original dataset release (v1.0) (Kang et al., 2026).[1] The subsequent v1.1 release applies semantic filtering to the evaluation split; scores from the two releases are not directly comparable. After merging and deduplication, the 35,059-skill pool contains 34,384 from the `skills-34k` collection in Shiyu-Lab's Skill-Usage dataset (Liu et al., 2026b), 13 from OpenClaw (OpenClaw contributors, n.d.), 77 from the Claude Code Skills Marketplace (Daymade, n.d.), and 585 from the production collection of OpenAI Plugins (OpenAI, n.d.). The Skill-Usage documents and accompanying resources are used as released. All methods search the same pool within each evaluation; the three task benchmarks use the 35,059-skill pool.

[1] The original release is available at https://huggingface.co/datasets/ThakiCloud/SKILLRET/tree/v1.0.

A.3 RETRIEVER TRAINING

The shared Qwen3-Embedding-8B encoder is adapted with rank-16 LoRA for one epoch on the 260,631 matched pairs. Training uses a learning rate of 2×10^{-5}, an effective batch of 80, and temperature $\tau = .05$ in Eq.(3). The objective follows the SentenceTransformers multiple-negatives implementation (Reimers & Gurevych, 2019). Query and document representations share the adapters. Training inputs and skill documents are limited to 8,192 tokens; online query inputs are limited to 512.

Implementation. Retriever training uses BF16 with PyTorch 2.10.0, CUDA 12.8, Sentence Transformers 5.6.0, Transformers 5.12.1, PEFT, and Ray. Local Qwen inference uses vLLM on NVIDIA H20 GPUs, and dense retrieval uses FAISS. DeepSeek-V4-Flash is accessed through its model API.

B CONTEXTUAL SKILL SELECTION

Each executor uses its own base model for online Dream generation, applicability assessment, and final selection. The independent assessments read one candidate at a time. They use only observations already available to the executor; the selector performs no internal terminal checks and does not pre-install candidates. When several candidates pass, the model compares their full bodies and resources in the same execution context. The final instruction asks it to:

> Choose the reference whose original guidance best supports the task and observed state.

It selects one of the accepted candidates. The empty and singleton cases follow Eq.(6) directly.

Skill activation. A successful search activates the selected skill and returns its activation status; no selection returns NONE. From the next turn, the system message includes an Active Skill / Reference text block containing the guidance and resource entry points. Activated skills accumulate across searches. Repeated activation reuses existing resources, and NONE preserves prior activations. The original task remains in the user message.

Original guidance and resources. For DreamSkill, YAML frontmatter is removed from the delivered body. Bodies up to 5,000 tokens are supplied in full. For longer bodies, the executor's base model selects original sections, which are inserted in source order within the budget rather than rewritten. Public baselines deliver complete selected bodies without this section-selection step; multiple selected skills are delivered individually.

After selection, the complete skill bundle is copied to a dedicated sandbox directory, preserving scripts, references, assets, and registered dependency directories. The Resources block gives the skill root and relative file paths for the executor to read or invoke through its existing terminal. Activation neither runs scripts automatically nor registers new tools.

Delivered context. The system message orders the executor instructions, skill blocks, native-environment description, and current terminal view. DreamSkill supplies the selected source text, not selector reasoning, evidence-section identifiers, or generated usage advice. Baseline reference wrappers include a query-derived *Suggested use* field. Subsequent searches and actions follow Appendix C.1.

C EVALUATION AND SKILL PROVENANCE

C.1 EXECUTION AND SAMPLING

For the three task benchmarks, all methods use a unified agent loop built on OpenEnv (Spisak et al., 2025). Each rollout runs in an independent Linux sandbox with a persistent terminal, a benchmark-specific executor prompt, and the benchmark's native verifier. Skill-enabled methods obtain their first `search_skill` result before normal task execution. They may first make one read-only terminal observation, after which search is required; No Skill does not invoke retrieval.

A final response without a tool call ends the rollout. Each generation is capped at 32,768 output tokens. For Spider2-DBT, we exclude four projects with upstream data issues (`chinook001`, `biketheft001`, `google_ads001`, and `gitcoin001`), following the public Harbor adapter, and evaluate all methods on the remaining 64 tasks.[2]

Search queries. The executor's `query` argument describes the requested deliverable, relevant observed facts, and the reusable method or decision sought. Methods run independently, so their online queries need not be identical. The U, F, and D variants retain C's Query+Dream construction, B omits Dream from the embedding input, and selection variants retain C's retrieval input construction. The cached-Dream controls in Appendix C.5 apply to static retrieval evaluation. Exact search instructions and input templates appear in Appendix E.

Executor decoding. Qwen3.5-9B and Qwen3.5-27B enable thinking with an 8,192-token thinking budget. They use temperature 0.6, top-p 0.95, top-k 20, min-p 0, presence penalty 0, and repetition penalty 1. DeepSeek-V4-Flash enables thinking with reasoning effort set to `high`; temperature and top-p are not explicitly overridden. Seeds are derived from the benchmark, task, trial, model role, and call index within that role, without including the method name.

C.2 Matched-guidance study

The study uses 20 warehouse analytics and reporting tasks from Toolathlon-GYM (Zhang et al., 2026), which builds on Toolathlon (Li et al., 2026b). The same Qwen3.5-9B executor runs five attempts per task under each of two conditions, totaling 200 measured attempts. The mean-score gain is 20.00 percentage points (paired 95% CI: $[9.00, 30.00]$). No Skill adds no procedural guidance. Matched Skill directly supplies manually selected public skills in their original form. Table 7 lists the seven skills used across these tasks. Their names and sources describe this motivating intervention; the main benchmarks evaluate automatic retrieval and selection from the 35,059-skill pool.

Table 7: Public skills used in the matched-guidance study and their source references.

Source	Skills used
Knowledge Work Plugins (Anthropic, n.d.)	`sql-queries`, `validate-data`
Hermes Agent (Nous Research, n.d.)	`xlsx`, `docx`
Claude Skills (Jeffallan, n.d.)	`pandas-pro`
ZJUNLP Skills (ZJUNLP, n.d.)	`multi-source-web-researcher`
SkillsBench (Li et al., 2026c)	`data-cleaning`

C.3 Baseline identities

Table 8 identifies the models used by the task baselines.

Table 8: Retrieval models and downstream selection used by the task baselines.

Task method	Embedding / lexical model	Reranker or selector
BM25	Lexical BM25	None
Qwen3 retrieval	Qwen3-Embedding-8B	Qwen3-Reranker-0.6B
SkillRet	SKILLRET-Embedding-8B	Qwen3-Reranker-0.6B
R3-Skill	R3-embedding-0.6b	R3-rerank-0.6b
SkillRouter	SkillRouter-Embedding-0.6B	SkillRouter-Reranker-0.6B
SkillReason	SkillReason-embedding-4b	SkillReason-reranker-0.6b
SkillFlow	bge-base-en-v1.5	Two cross-encoders; LLM selection
DreamSkill	DreamSkill-Embedding-8B	Executor-based contextual selection

[2] `https://github.com/harbor-framework/harbor/blob/main/adapters/spider2-dbt/README.md`

Baseline inputs and processing follow each method's original procedures. BM25 delivers its Top-1 skill. Qwen3 retrieval, SkillRet, R3-Skill, SkillRouter, and SkillReason retrieve Top-20 candidates, apply their corresponding rerankers, and deliver only the Top-1 skill. SkillFlow generates five retrieval queries, retrieves 200 candidates per query, merges them, and applies two reranking stages. Its selector considers the final five candidates and delivers every selected skill, yielding zero to five skills. LLM-based query generation and selection use the corresponding executor's base model. Baselines deliver original bodies and resources as described in Appendix B. Retrieval-only comparisons evaluate lexical or embedding ranking before reranking and selection, including bge-base-en-v1.5 (Xiao et al., 2024), the encoder used by SkillFlow.

C.4 Task outcomes and repeated attempts

For task i and trial r, let $b_{ir} \in \{0, 1\}$ indicate verifier success. SpreadsheetBench and Spider2-DBT require reward exactly one; Terminal-Bench uses positive verifier reward. Episodes without a valid verifier outcome count as unsuccessful; a valid outcome is retained even when an episode times out. For N tasks and n trials,

$$\text{Accuracy/SR} = \frac{100}{Nn} \sum_{i=1}^{N} \sum_{r=1}^{n} b_{ir}. \tag{7}$$

Pass@k is the percentage of tasks solved at least once in the first k trials. The main benchmarks use three trials per task; the matched-guidance study uses five and averages its score over all 100 attempts per condition. Table 1 reports gains in percentage points over No Skill.

C.5 Retrieval usefulness and label checking

Static retrieval inputs. We evaluate 6,460 capability queries derived from the SkillRet test split on the two skill pools. Qwen3.5-9B generates each Dream offline with thinking disabled, taking only the derived capability query as input; it does not read gold annotations or candidate skills. Dreams are cached after generation. C, U, D, and F share the same cached Dream for each query, while B removes Dream from the input without regenerating it. These controls apply to static retrieval evaluation and ablations; during task execution, each executor generates its own Dreams as described in Section 3.4.

Usefulness judgment. Qwen3-32B reads a query and one candidate's name, description, and body. A skill is useful when it offers concrete steps, a decision or validation rule, or another procedure that supports the query or an explicit constraint. Useful partial guidance qualifies; topic overlap alone does not. The judgment permits ordinary adaptation to the request. The judge does not receive Dream, source gold labels, retrieval scores, or the online execution state. Long candidate bodies are truncated to its input budget, and resources are visible when described in the candidate text.

Let $y_{qi} = 1$ when the skill at rank i for query q is useful, and let Q contain all 6,460 evaluation queries. Define $r_q = \min\{i \le 10 : y_{qi} = 1\}$ when a useful result exists. Then

$$\text{JU@}k = \frac{100}{|Q|} \sum_{q \in Q} \mathbf{1}\left[\sum_{i=1}^{k} y_{qi} > 0\right], \tag{8}$$

$$\text{MRR@10} = \frac{1}{|Q|} \sum_{q \in Q} \begin{cases} 1/r_q, & \text{a useful skill exists in Top-10,} \\ 0, & \text{otherwise} \end{cases}, \tag{9}$$

$$\text{P@}k = \frac{100}{|Q|k} \sum_{q \in Q} \sum_{i=1}^{k} y_{qi}. \tag{10}$$

JU measures access to at least one useful skill, MRR measures its rank, and P measures the useful fraction of the candidate list. These are retrieval judgments, distinct from verifier-based task success.

We checked 200 distinct query–skill pairs, sampled from the pooled Top-10 outputs of the compared retrievers, with 100 from each pool. Two annotators independently applied the binary usefulness criterion to the query and candidate text, with model identity, rank, Dream, and automatic labels

hidden. They agreed on 194 pairs (97.00%) and jointly resolved the six disagreements. The final labels contain 112 useful and 88 non-useful pairs. Against these labels, the automatic judge achieves 86.50% accuracy, 84.55% precision, 92.86% recall, and 88.51% F1, with useful as the positive class.

D Complete Ablations and Uncertainty

The full-model reference is C throughout. Task ablations use Qwen3.5-9B, the 35,059-skill pool, and complete episodes from the same initial tasks. F, U, B, and D retain the full contextual selector. Retrieval comparisons use each model to encode its query input and the skill documents.

Task effects pair methods on the same tasks and resample tasks as clusters, retaining all attempts within each task. SD is the sample standard deviation across the three trial-level scores. Retrieval effects use 10,000 paired bootstrap resamples of the 1,006 source-query groups (seed 20260828), retaining each group's derived needs. All intervals are paired 95% bootstrap confidence intervals; each effect is the complete model minus its reference.

D.1 Retrieval learning and representation

F uses the original Qwen3-Embedding-8B checkpoint with Query+Dream and full skill documents. C uses the trained retriever with the same input format and downstream decisions. Their comparison measures the contribution of retrieval learning.

U preserves C's retained queries, Dreams, and training recipe but replaces the matched positive for each need with one uniformly sampled skill from the source task's gold set. The sample is drawn once and fixed before training; it may equal C's matched skill. U starts from the same original backbone and uses the single-positive objective in Eq.(3), with the sampled skill in place of s_i^+. Thus C/U changes the assignment of training supervision.

B and D use C's matched training pairs and training recipe. B omits Dream from the query input at training and retrieval. D retains Query+Dream but encodes skill names and descriptions instead of full bodies. Their comparisons with C measure the query and document representations, respectively.

Table 9: Complete retrieval ablations on both skill pools. F uses Qwen3-Embedding-8B; C is the full trained retriever. U changes positive assignment, B omits Dream, and D encodes metadata. JU and P are percentages; MRR@10 is on $[0, 1]$.

Pool size: 6,660 skills Variant	JU@1	JU@5	JU@10	MRR@10	P@5	P@10
Qwen3-Embedding-8B (F)	77.32	96.11	97.88	.8558	64.06	57.50
B: Query-only input	81.33	97.34	98.76	.8847	68.88	62.57
D: Metadata embedding	77.76	96.42	98.36	.8609	64.07	58.34
U: w/o step-level matching	78.72	96.93	98.56	.8664	64.80	58.60
C: DreamSkill-Embedding-8B	81.80	97.54	98.81	.8886	69.03	62.97
Pool size: 35,059 skills Variant	JU@1	JU@5	JU@10	MRR@10	P@5	P@10
Qwen3-Embedding-8B (F)	76.18	95.68	97.80	.8477	69.38	65.38
B: Query-only input	76.95	96.84	98.45	.8573	71.76	68.47
D: Metadata embedding	73.59	95.42	97.82	.8318	67.50	64.34
U: w/o step-level matching	70.68	95.50	97.93	.8143	66.49	63.23
C: DreamSkill-Embedding-8B	77.35	97.11	98.58	.8605	71.99	68.83

C leads all six retrieval metrics across these variants on both pools (Table 9), including JU@10 and the useful fraction measured by P@5 and P@10. The paired effects in Table 10 connect these improvements to the learning design. Need-specific positive assignment raises MRR@10 over U by .0223 and .0462 on the 6,660- and 35,059-skill pools, respectively, with both intervals above zero. Compared with the original checkpoint F, C improves JU@5 and MRR@10 on both pools. The C/D comparisons favor full-document encoding across JU@1, JU@5, and MRR@10. These results

Table 10: Paired retrieval effects of C relative to U, F, B, and D: differences [95% CI]. Variants follow Table 9. JU differences are percentage points; MRR@10 uses $[0, 1]$.

Contrast	ΔJU@1	ΔJU@5	ΔMRR@10
Pool size: 6,660 skills			
C−U	+3.08 [2.23, 3.91]	+0.60 [0.28, 0.94]	+0.0223 [0.0173, 0.0272]
C−F	+4.47 [3.34, 5.59]	+1.42 [0.96, 1.89]	+0.0328 [0.0260, 0.0397]
C−B	+0.46 [−0.23, 1.16]	+0.20 [−0.08, 0.48]	+0.0039 [−0.0001, 0.0078]
C−D	+4.04 [3.05, 5.01]	+1.11 [0.72, 1.51]	+0.0277 [0.0218, 0.0336]
Pool size: 35,059 skills			
C−U	+6.67 [5.67, 7.71]	+1.61 [1.18, 2.05]	+0.0462 [0.0401, 0.0525]
C−F	+1.18 [−0.08, 2.46]	+1.42 [0.91, 1.95]	+0.0127 [0.0048, 0.0206]
C−B	+0.40 [−0.48, 1.27]	+0.26 [−0.08, 0.59]	+0.0032 [−0.0018, 0.0082]
C−D	+3.76 [2.56, 4.96]	+1.69 [1.21, 2.16]	+0.0287 [0.0214, 0.0360]

Additional JU@10 effects for the 35,059-skill pool are +0.12 [−0.06, 0.31] for C−B and +0.76 [0.46, 1.07] for C−D, in percentage points.

support learning the correspondence to a supporting procedure and retaining its body in the retrieval representation.

D.2 Task ablations

The four selection variants retain C's trained retriever, executor, native tools, and reference-delivery rule. Embedding Top-1 directly adopts the first retrieved skill. Joint selection + NONE reads all Top-10 full documents with the execution context in one call, returning a candidate or NONE. Metadata-only selection removes only the skill body, retaining C's other inputs and decision procedure. Forced selection requires adopting a full candidate instead of returning NONE. Table 6 gives their scores; Table 11 combines the retriever-variant scores with paired gains for all eight ablations.

Table 11: Qwen3.5-9B task ablations. (a) Complete retriever-variant scores (%), with best values and ties in bold. (b) Paired C−variant Accuracy/SR gains [95% CI], in percentage points.

(a) Task scores

	SpreadsheetBench		Terminal-Bench		Spider2-DBT	
Variant	Acc.	Pass@3	Acc.	Pass@3	SR	Pass@3
F: Original checkpoint + Dream	39.83	58.00	28.09	**44.94**	14.58	25.00
B: Query-only input	37.00	57.00	**28.46**	42.70	13.02	21.88
D: Metadata	38.50	58.50	28.09	39.33	13.02	20.31
U: w/o step-level matching	38.42	58.50	27.34	38.20	17.19	28.13
C: DreamSkill	**41.58**	**59.25**	**28.46**	40.45	**18.75**	**32.81**

(b) Paired gains of C over each variant

Reference variant	SS	TB	DBT
U: w/o step-level matching	+3.17 [0.58, 5.75]	+1.12 [−3.75, 5.99]	+1.56 [−4.69, 7.81]
F: Original checkpoint + Dream	+1.75 [−1.00, 4.50]	+0.37 [−4.12, 4.49]	+4.17 [−0.52, 9.38]
B: Query-only input	+4.58 [2.00, 7.25]	0.00 [−5.62, 5.62]	+5.73 [1.04, 10.94]
D: Metadata encoding	+3.08 [0.42, 5.83]	+0.37 [−4.49, 5.24]	+5.73 [0.52, 10.94]
Embedding Top-1	+6.00 [3.25, 8.75]	+4.12 [−1.87, 10.49]	+3.65 [−1.04, 9.38]
Joint selection + NONE	+1.25 [−1.50, 4.08]	+1.12 [−4.12, 6.37]	+3.65 [−1.04, 8.85]
Metadata-only selection	+5.08 [2.42, 7.75]	+5.24 [−0.75, 11.61]	+4.69 [0.52, 9.38]
Forced selection	+3.08 [0.25, 6.00]	+1.50 [−3.75, 6.74]	+2.08 [−3.13, 7.81]

SS, TB, and DBT follow Table 2. Selection-variant scores appear in Table 6.

Retaining Dream improves task performance after the positive assignments have been fixed. Relative to B, C gains 4.58 percentage points on SpreadsheetBench and 5.73 on Spider2-DBT, with intervals $[2.00, 7.25]$ and $[1.04, 10.94]$ (Table 11). Full-document encoding yields the same 5.73-point Spider2-DBT gain over D. At the adoption stage, inspecting the full guidance improves Spider2-

DBT SR by 4.69 points over metadata-only selection, with interval $[0.52, 9.38]$. The two document comparisons concern different decisions: retrieving a procedure and choosing whether to use it.

D.3 MAIN-RESULT COMPARISONS

Tables 12 and 13 report the nine comparisons with No Skill and thirteen with the highest-scoring skill baselines, including all ties. The first comparison measures the benefit of acquiring guidance; the second compares our acquisition method with alternative skill pipelines.

Table 12: Task Accuracy/SR relative to No Skill. Means and sample SDs are percentages, with SD computed across three trials. Paired differences and 95% confidence intervals are in percentage points.

Benchmark	Ours ± SD	No Skill ± SD	Δ [95% CI]
Qwen3.5-9B			
SpreadsheetBench	41.58 ± 0.29	34.25 ± 0.90	$+7.33$ [4.25, 10.42]
Terminal-Bench	28.46 ± 3.43	21.35 ± 2.97	$+7.12$ [1.12, 13.11]
Spider2-DBT	18.75 ± 2.71	13.54 ± 1.80	$+5.21$ [0.00, 10.42]
Qwen3.5-27B			
SpreadsheetBench	68.42 ± 0.76	63.83 ± 2.10	$+4.58$ [1.42, 7.75]
Terminal-Bench	37.45 ± 1.30	35.58 ± 1.72	$+1.87$ [−4.12, 8.24]
Spider2-DBT	31.25 ± 1.56	30.73 ± 2.39	$+0.52$ [−4.69, 5.73]
DeepSeek-V4-Flash			
SpreadsheetBench	88.33 ± 2.01	82.42 ± 1.01	$+5.92$ [3.83, 8.08]
Terminal-Bench	68.16 ± 5.07	63.30 ± 3.24	$+4.87$ [0.00, 10.11]
Spider2-DBT	56.25 ± 1.56	44.79 ± 3.25	$+11.46$ [5.73, 17.71]

Table 13: Paired Accuracy/SR gains over the highest-scoring skill baselines in each setting. All tied baselines are included. Baseline scores are mean ± SD (%); differences and 95% confidence intervals are in percentage points. Ours scores appear in Table 12.

Benchmark	Reference	Reference ± SD	Δ [95% CI]
Qwen3.5-9B			
SpreadsheetBench	SkillRouter	37.92 ± 2.57	$+3.67$ [0.92, 6.42]
Terminal-Bench	BM25	22.47 ± 2.25	$+5.99$ [0.00, 11.99]
Terminal-Bench	SkillRet	22.47 ± 2.97	$+5.99$ [1.50, 10.86]
Spider2-DBT	SkillRet	15.63 ± 4.69	$+3.13$ [−1.56, 7.81]
Qwen3.5-27B			
SpreadsheetBench	SkillFlow	66.92 ± 2.02	$+1.50$ [−1.08, 4.08]
Terminal-Bench	SkillFlow	35.96 ± 1.95	$+1.50$ [−4.12, 7.12]
Spider2-DBT	SkillRet	30.73 ± 5.92	$+0.52$ [−5.21, 6.25]
DeepSeek-V4-Flash			
SpreadsheetBench	R3-Skill	83.92 ± 1.23	$+4.42$ [2.58, 6.25]
Terminal-Bench	SkillRet	64.04 ± 1.95	$+4.12$ [−1.12, 9.36]
Terminal-Bench	R3-Skill	64.04 ± 3.37	$+4.12$ [−1.12, 9.74]
Spider2-DBT	SkillFlow	54.17 ± 0.90	$+2.08$ [−2.08, 6.77]
Spider2-DBT	SkillRet	54.17 ± 3.25	$+2.08$ [−2.08, 6.25]
Spider2-DBT	SkillReason	54.17 ± 2.39	$+2.08$ [−1.56, 5.73]

Against No Skill, DreamSkill improves mean Accuracy/SR in all nine settings (Table 12). SpreadsheetBench gains are positive for each executor, with intervals above zero. The DeepSeek-V4-Flash result on Spider2-DBT shows an 11.46-point gain with interval $[5.73, 17.71]$ while reusing the same trained retriever.

Against the strongest skill baselines (Table 13), Qwen3.5-9B gains 3.67 points over SkillRouter on SpreadsheetBench and 5.99 over each tied Terminal-Bench baseline. DeepSeek-V4-Flash gains 4.42 points over R3-Skill on SpreadsheetBench, with interval $[2.58, 6.25]$. These comparisons place the task gains relative to both an unguided executor and existing ways of acquiring skills, with trial variation and paired precision reported alongside the means.

E PROMPTS AND INPUT FORMATS

Initial search instruction.

> Before normal execution, obtain one SearchSkill result. If the task already provides enough concrete information, you may search immediately. Otherwise, use the terminal once to inspect the workspace or target artifact, then search on the next turn. The optional pre-search terminal call is for inspection rather than modification. Use its observation to make the search query specific. SearchSkill may return NONE.

Search tool. The tool is named `search_skill` and has one required string argument, `query`. Its description is:

> Search the skill corpus for reusable instructions that can help with the current task, or return NONE.

The `query` parameter description is:

> Describe the requested deliverable, the relevant facts already observed, and the reusable method or decision that would change how the deliverable is produced. Keep the task outcome primary; do not replace it with an intermediate operation.

Online Dream generation. The system prompt is:

> Rewrite the search request as a compact excerpt that could appear in a matching skill. Describe the reusable capability, essential operations or technical objects, and the stated constraints or result in one or two concise sentences. Use only facts stated in the request and keep implementation choices exactly as specified there. Return only the excerpt.

The user message contains only the search query. The returned excerpt is used directly as Dream text.

Query+Dream embedding input. The full query-side input for C is:

```
Instruct: Given a short skill-search query and a
compact hypothetical skill description, retrieve skill
documents that provide the requested capability
Query: Skill-search query:
{query}
Hypothetical skill:
{dream_text}
```

Applicability and selection inputs. Each applicability call receives one candidate's name, description, frontmatter, body annotated with section identifiers, and resource file list. Both stages receive the original task, current terminal view, search query, and native-tool names and descriptions in the `<task>`, `<observed_state>`, `<search_query>`, and `<native_tools>` fields, respectively. The final selection call receives the original texts and metadata of all candidates accepted by the independent applicability judgments.