IMAGINE BEFORE ACTING: PROSPECTIVE SKILL IDENTIFICATION FOR LLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Effective skill identification is crucial for LLM agents to reuse task-solving experience for complex task execution, which however remains challenging as agents must accurately identify useful skills from a large skill gallery and invoke them on demand. Existing studies have explored various strategies for skill retrieval and invocation, but largely overlook the natural discrepancy between task descriptions and skill documents. To be specific, our empirical study reveals that task descriptions mainly specify task objectives, whereas skill documents describe the required capabilities and procedural guidance for achieving the objectives. Such a so-called Task–Skill Misalignment (TSM) problem would cause desirable skills to be neglected during retrieval or remain unselected during invocation, thus hindering task execution. As a remedy, inspired by human prospective cognition, we propose SkillDreamer, a prospective skill identification framework to mitigate TSM. In brief, SkillDreamer first infers execution-required capabilities and imagines their realizations as pseudo skills, thereby translating task objectives into execution-oriented guidance for skill retrieval. As execution proceeds, SkillDreamer reevaluates the capabilities required by emerging subtasks to determine which skills should be invoked, thereby aligning skill invocation with evolving execution needs. Extensive experiments on three representative agent skill benchmarks not only verify the effectiveness of SkillDreamer in mitigating TSM, but also demonstrate its generalizability across diverse retrievers and LLM agents. The code will be released upon acceptance.

1 INTRODUCTION

Large language model (LLM) agents have emerged as a promising paradigm for solving complex tasks through autonomous planning (Yao et al., 2023), external tool invocation (Schick et al., 2023), and dynamic interaction with environments (Wang et al., 2023). However, solving each incoming task from scratch requires agents to repeatedly devise execution plans and rely solely on their inherent capabilities, limiting both execution efficiency and reliability. As a remedy, Anthropic (2025) proposes encapsulating task-specific workflows, domain expertise, and supporting resources into structured documents, termed *Skills*, whereby agents could access and reuse task-solving knowledge on demand. As human-authored and automatically generated skills rapidly accumulate in large skill libraries, identifying the most useful skills for given tasks becomes increasingly challenging, making effective skill identification as one of the most fundamental challenges for agents.

Toward effective skill identification, existing agents generally follow a two-stage paradigm, *i.e.*, skill retrieval and skill invocation. Specifically, given an overall task description, the agent first retrieves a set of candidate skills from the skill gallery to support subsequent execution. As execution proceeds, the agent progressively plans and decomposes the overall task into concrete subtasks, invoking relevant skills from the candidate set on demand. Within this paradigm, prior studies have explored different strategies for skill retrieval and invocation. On the one hand, during skill retrieval, existing methods focus on improving retrieval performance by training skill-customized retrievers (Cho et al., 2026), adopting multi-stage retrieve-and-rerank pipelines (Zheng et al., 2026), or decomposing complex tasks for finer-grained matching (Chen et al., 2026c; Gao, 2026). On the other hand, during skill invocation, existing methods aim to improve the accuracy of selecting desirable skills by resorting to the current environment state and interaction history (Wang et al., 2025; Li et al., 2026; Chen et al., 2026b), or learn policies for whether and when to invoke skills (Chen et al., 2026a).

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.

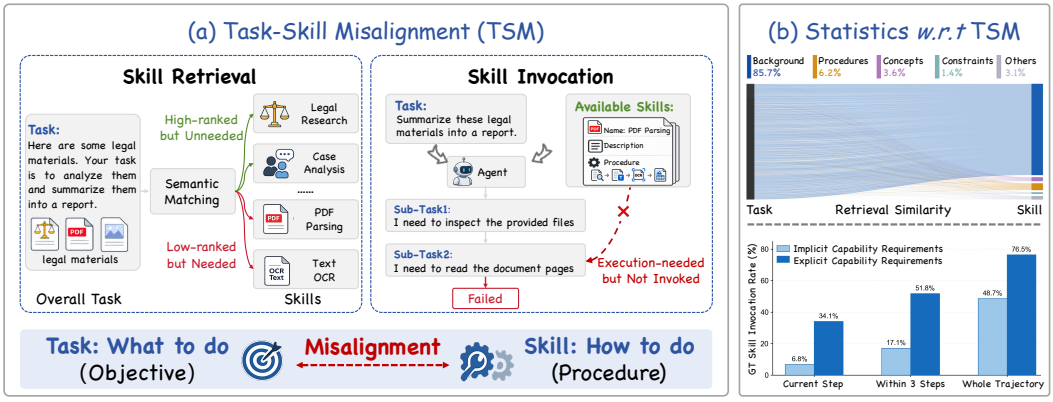


Figure 1: **(a) Our observation.** Task–Skill Misalignment: tasks mainly specify desired objectives, while skills usually manifest as the procedural knowledge required for execution. As a result, execution-critical skills might be ranked lower during retrieval or remain uninvoked during execution. **(b) Statistics *w.r.t* TSM.** Top: task–skill matching is dominated by background information, while execution-related procedures contribute much less. Bottom: explicit capability requirements are associated with higher invocation rates of ground-truth skills. In this figure, we use Qwen3-Embedding-4B to measure the similarity between the query task and different components of skills on the SkillRet benchmark. For invocation, we provide DeepSeek-V4-Flash with ground-truth skills on SkillUsage and use GPT-6-Sol to evaluate whether capability requirements are explicitly expressed in the execution trajectories.

Despite their promising performance, existing studies always neglect the gap between task descriptions and skill documents, leading to the so-called Task–Skill Misalignment (TSM) problem. To be specific, as illustrated in Fig. 1(a), task mainly specify the objectives of users (*e.g.*, generating reports) whereas skills typically describe the capabilities and execution guidance required to achieve the objectives (*e.g.*, PDF Parsing). As shown in Fig. 1(b), the TSM problem could hinder an incentive to significantly hinder the correct identification for desirable skill during both the skill retrieval and invocation stages. To bridge this gap, we draw inspiration from human *prospective cognition*, which involves imagining future situations in advance to guide current decisions (Gilbert & Wilson, 2007). Similarly, effective skill identification requires looking beyond task objectives to imagine what execution will demand. Therefore, we aim to reason about the capabilities needed to accomplish a task or subtask and render these requirements explicit, bridging the gap between task objectives and execution-oriented skills across both retrieval and invocation.

Based on the above observations and motivations, we propose SkillDreamer, a prospective skill-identification framework for mitigating TSM across skill retrieval and invocation. In brief, SkillDreamer consists of three modules, *i.e.*, Prospective Skill Generation (PSG), Hybrid Skill Retrieval (HSR), and Capability-aware Skill Invocation (CSI). Given a task, PSG first infers the required capabilities and imagines how they could be realized, formulating the anticipated procedural guidance as pseudo skills. After that, HSR combines these pseudo skills with the original task description to retrieve candidate skills, accounting for both task relevance and execution needs. During execution, CSI continually reevaluates the capabilities required by emerging subtasks and determines whether and which candidate skills to invoke.

In summary, the main contributions and novelties of this work could be summarized as follows:

- We study a practical yet less-explored prblem termed Task–Skill Misalignment (TSM) in agent skill identification, and empirically reveal that the discrepancy between task requirements and execution-oriented skills could significantly hinder both skill retrieval and invocation.
- Inspired by prospective cognition of humans, we propose SkillDreamer, a prospective skill-identification framework that anticipates required capabilities and their realization strategies for retrieval, while further performing capability-aware skill invocation during execution.

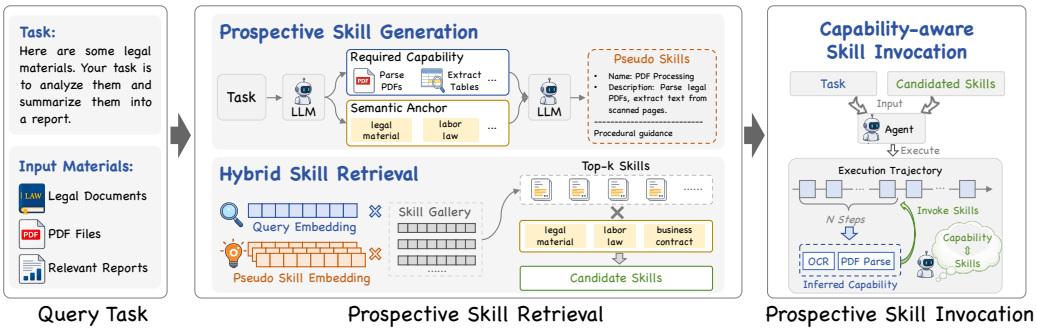


Figure 2: **Overview of the proposed SkillDreamer.** Given a task, PSG anticipates the capabilities required for execution and generates corresponding pseudo skills while preserving task-specific semantics. HSR then combines the original task with the generated pseudo skills to retrieve a set of candidate skills. During execution, CSI reassesses the capability required by the current subtask before action and invokes an appropriate candidate skill when needed.

- We propose SkillDreamer, a novel framework which unifies skill retrieval and invocation through prospective reasoning over explicit capability requirements. By translating task objectives into imagined procedural guidance for retrieval and exposing emerging capability needs for invocation, SkillDreamer could bridge task-side intent and skill-side execution knowledge across both stages.
- Extensive experiments on three representative skill benchmarks verify the effectiveness of SkillDreamer in both skill retrieval and end-to-end task execution. Further evaluations across multiple retrievers and LLM agents establish its broad applicability.

2 METHOD

In this section, we present SkillDreamer for mitigating Task–Skill Misalignment (TSM) in skill retrieval and invocation. As illustrated in Fig. 2, SkillDreamer consists of two stages, namely, Prospective Skill Retrieval and Prospective Skill Invocation. The section is organized as follows. In Sec. 2.1, we first formally define TSM in both stages. In Sec. 2.2, we introduce Prospective Skill Generation (PSG) to infer task-required capabilities and construct corresponding pseudo skills. In Sec. 2.3, we present Hybrid Skill Retrieval (HSR) to retrieve candidate skills by jointly considering the original task and prospective execution information. Finally, in Sec. 2.4, we introduce Capability-aware Skill Invocation (CSI) to identify capability requirements during execution and invoke appropriate skills on demand.

2.1 PROBLEM FORMULATION

Given a task q and a skill gallery $\mathcal{S} = \{s_i\}_{i=1}^{G}$, agent skill identification generally involves two stages, *i.e.*, skill retrieval before execution and skill invocation during execution. At the retrieval stage, a retriever $\mathcal{R}$ ranks the gallery skills according to the overall task and selects the top-K candidates for subsequent execution:

$$\mathcal{C} = \mathcal{R}(q, \mathcal{S}), \qquad |\mathcal{C}| = K, \tag{1}$$

where $\mathcal{C}$ denotes the retrieved candidate skill set.

During task execution, the agent progressively plans and decomposes the overall task into a sequence of concrete subtasks. For example, generating a report from legal documents might involve parsing the source files, extracting relevant evidence, and composing the final report. We denote the subtask sequence formed up to execution stage t as

$$\mathcal{T}_t = (\tau_1, \ldots, \tau_t), \tag{2}$$

where τ_t denotes the current subtask. Based on the current subtask sequence and the retrieved candidate skills, the invocation process $\mathcal{I}$ determines the skill to be invoked:

$$s_t^{\text{inv}} = \mathcal{I}(\mathcal{T}_t, \mathcal{C}), \qquad s_t^{\text{inv}} \in \mathcal{C} \cup \{\varnothing\}, \tag{3}$$

where $s_t^{\text{inv}} = \varnothing$ indicates that the agent proceeds without invoking any candidate skill.

As discussed in the Introduction, Task–Skill Misalignment (TSM) might arise at both stages of skill identification. We first characterize its manifestation in skill retrieval.

Definition 1 (Retrieval-side Task–Skill Misalignment) *Given a task q, let s^+ denote a useful skill for task execution and s^- denote a less useful skill. Retrieval-side TSM occurs when*

$$r(q, s^+) \leq r(q, s^-), \tag{4}$$

where $r(q, s)$ denotes the retrieval score between task q and skill s.

In this case, an execution-useful skill receives lower retrieval priority than a less useful one and might consequently be excluded from the candidate set $\mathcal{C}$. Moreover, TSM might persist as follows even when the required skill has already been successfully retrieved.

Definition 2 (Invocation-side Task–Skill Misalignment) *For a subtask τ_t, let $s_t^* \in \mathcal{C}$ denote the skill required to accomplish it. Invocation-side TSM occurs when the required skill is available but not invoked, i.e.,*

$$s_t^* \in \mathcal{C}, \qquad s_t^{\text{inv}} \neq s_t^*. \tag{5}$$

Based on the above formulation, TSM manifests as two complementary failures: execution-critical skills might be under-prioritized during retrieval or remain uninvoked during execution. To address both, we instantiate the prospective reasoning introduced in the Introduction at the retrieval and invocation stages, respectively. In the following, we elaborate on the corresponding designs.

2.2 Prospective Skill Generation

As discussed above, the task objective might not explicitly reveal the capabilities required for successful execution, making direct task–skill matching prone to overlooking execution-critical skills. To remedy this, we introduce Prospective Skill Generation (PSG), which first uncovers the capability requirements underlying the task and then prospectively imagines how these capabilities could be realized.

Specifically, given a task q, we employ an off-the-shelf LLM $\mathcal{M}$ to infer a set of capability requirements $\mathcal{U} = \{u_j\}_{j=1}^{M}$. Different from simply decomposing the explicit task description into finer-grained subqueries, PSG further reasons about the implicit capabilities required for task execution. Accordingly, each u_j characterizes an independent capability that requires specialized skill guidance, while trivial operations that could be readily handled by the agent itself are excluded.

Although capability inference could uncover the execution requirements underlying the task, reformulating the original task into capability-level descriptions might discard task-specific semantic information. Such contextual information remains important for distinguishing skills that provide similar capabilities but target different application scenarios. To preserve such information, PSG further extracts a set of semantic anchors $\mathcal{A} = \{a_l\}_{l=1}^{L}$ from the original task, including informative cues such as domain terminology, key objects, and task targets. Accordingly, the capability inference process is formulated as

$$(\mathcal{U}, \mathcal{A}) = \mathcal{M}\left(q; \mathcal{P}_{\text{PSG}}^{\text{inf}}\right), \tag{6}$$

where $\mathcal{P}_{\text{PSG}}^{\text{inf}}$ denotes the capability inference prompt.

Although $\mathcal{U}$ reveals the capabilities required for task execution, such concise capability descriptions still provide limited information about how these capabilities could be realized in practice. As a result, a discrepancy remains between capability-level descriptions and real skills that provide detailed execution guidance. To further bridge this discrepancy, PSG performs prospective reasoning to imagine the execution guidance that a useful skill would provide for each inferred capability. Specifically, for each $u_j \in \mathcal{U}$, PSG generates a corresponding pseudo skill as

$$\tilde{s}_j = \mathcal{M}\left(u_j, \mathcal{A}; \mathcal{P}_{\text{PSG}}^{\text{gen}}\right), \qquad j = 1, \ldots, M, \tag{7}$$

where $\mathcal{P}_{\text{PSG}}^{\text{gen}}$ denotes the prospective generation prompt. Accordingly, the generated pseudo skill set is denoted as $\widetilde{\mathcal{S}} = \{\tilde{s}_j\}_{j=1}^{M}$. Rather than merely restating the inferred capability, each pseudo skill

is formulated in a skill-like form that describes how the corresponding capability could be realized through concrete procedural guidance.

In this way, PSG transforms implicit capability requirements into execution-oriented pseudo skills while preserving task-specific semantics. These pseudo skills thereby provide prospective retrieval cues that better align with the procedural knowledge encoded in real skills.

2.3 HYBRID SKILL RETRIEVAL

Although PSG provides skill-like descriptions of the anticipated execution requirements, the generated pseudo skills are imagined rather than verified and thus should not be directly treated as executable skills. Instead, we employ them as retrieval proxies and introduce Hybrid Skill Retrieval (HSR), which jointly considers the original task and generated pseudo skills to identify useful skills from the gallery.

To be specific, we first encode the task q, each pseudo skill $\tilde{s}_j \in \widetilde{\mathcal{S}}$, and each gallery skill $s_i \in \mathcal{S}$ using an embedding model $f(\cdot)$:

$$\mathbf{z}_q = f(q), \qquad \tilde{\mathbf{z}}_j = f(\tilde{s}_j), \qquad \mathbf{z}_i = f(s_i). \tag{8}$$

We then calculate the relevance between the original task and each gallery skill as

$$r_i^q = \mathrm{sim}\left(\mathbf{z}_q, \mathbf{z}_i\right), \tag{9}$$

where r_i^q measures the relevance of gallery skill s_i to the original task. Meanwhile, each pseudo skill is used as a retrieval proxy to identify gallery skills with similar capabilities and procedural guidance:

$$r_{ij}^p = \mathrm{sim}\left(\tilde{\mathbf{z}}_j, \mathbf{z}_i\right), \tag{10}$$

where r_{ij}^p measures the relevance between pseudo skill $\tilde{s}_j$ and gallery skill s_i.

To jointly account for task relevance and prospective execution information, we formulate the following fusion score:

$$d_{ij} = \alpha r_i^q + r_{ij}^p, \tag{11}$$

where α controls the contribution of the original task relevance. Unless otherwise specified, we set $\alpha = 2$. For each pseudo skill $\tilde{s}_j$, the top-K_c gallery skills ranked by d_{ij} are retained as the preliminary candidate set $\mathcal{B}_j$.

Although the above fusion incorporates both task and execution information, embedding-based matching might still underemphasize task-specific semantic cues preserved in the original task. To further retain such information, HSR employs the semantic anchors $\mathcal{A}$ extracted by PSG for BM25-based reranking within each candidate set $\mathcal{B}_j$. Let a denote the lexical query constructed by concatenating the semantic anchors in $\mathcal{A}$. For each candidate $s_i \in \mathcal{B}_j$, its anchor-based relevance is calculated as

$$r_i^a = \mathrm{BM25}\left(a, s_i\right), \qquad s_i \in \mathcal{B}_j. \tag{12}$$

Since the three relevance scores might have different numerical scales, we apply min-max normalization within each candidate set, yielding $\bar{r}_i^q$, $\bar{r}_{ij}^p$, and $\bar{r}_i^a$. The final reranking score is then formulated as

$$h_{ij} = \alpha \bar{r}_i^q + \bar{r}_{ij}^p + \bar{r}_i^a. \tag{13}$$

Each pseudo skill therefore produces a capability-specific ranking $\mathcal{L}_j$ according to h_{ij}. Since a complex task might involve multiple complementary capabilities, selecting skills from only one ranking could overlook part of the inferred capability requirements. We therefore retain the top-K_r skills from each ranking and organize them into a ranking matrix. Specifically, let $\pi_j(v)$ denote the index of the v-th ranked gallery skill in $\mathcal{L}_j$, and define

$$\mathbf{R} = \begin{bmatrix} s_{\pi_1(1)} & s_{\pi_2(1)} & \cdots & s_{\pi_M(1)} \\ s_{\pi_1(2)} & s_{\pi_2(2)} & \cdots & s_{\pi_M(2)} \\ \vdots & \vdots & \ddots & \vdots \\ s_{\pi_1(K_r)} & s_{\pi_2(K_r)} & \cdots & s_{\pi_M(K_r)} \end{bmatrix}, \tag{14}$$

where the M columns correspond to the inferred capability requirements and the K_r rows correspond to their ranking positions. Starting from the first row, HSR scans $\mathbf{R}$ from left to right before

proceeding to the next row, while skipping duplicate skills. The process terminates once K unique skills are collected, yielding the final candidate set

$$\mathcal{C} = \{\hat{s}_1, \hat{s}_2, \ldots, \hat{s}_K\}. \tag{15}$$

In this way, different capability requirements can contribute their highly ranked skills to the final candidate set, thereby improving the coverage of skills needed for subsequent execution.

2.4 Capability-aware Skill Invocation

Although HSR retrieves candidate skills $\mathcal{C}$ before execution, the current subtask may not explicitly reveal the capability required to accomplish its objective, causing an available execution-critical skill to remain uninvoked as detailed in the Introduction. To address such invocation-side TSM, we introduce Capability-aware Skill Invocation (CSI), which makes the implicit capability requirement explicit and uses it to guide skill invocation during execution.

Following Sec. 2.1, let $\mathcal{T}_t$ denote the subtask sequence formed up to execution stage t. During execution, the agent progressively accomplishes the current subtask by selecting and performing concrete actions. Before taking the next action, CSI reevaluate the current objective, infers the capability requirement u_t, and determines whether additional skill guidance is needed as follows:

$$\left(u_t, s_t^{\text{inv}}\right) = \mathcal{I}_{\text{CSI}}\left(\mathcal{T}_t, \mathcal{C}; \mathcal{P}_{\text{CSI}}, \mathcal{X}_n\right), \tag{16}$$

where $\mathcal{P}_{\text{CSI}}$ denotes the capability-aware invocation instruction and $\mathcal{X}_n$ denotes the runtime context introduced below. If a candidate skill provides the required capability, CSI invokes it to support the next action; otherwise, $s_t^{\text{inv}} = \varnothing$ and the agent proceeds without additional skill guidance. Accordingly, CSI performs skill invocation through an explicit task objective–capability–skill reasoning process instead of directly matching the current subtask with candidate skills. The detailed assessment format and prompts are provided in the supplementary materials.

During long-horizon execution, the capability-aware instruction may receive diminishing attention as the interaction context grows. To maintain consistent capability assessment, CSI further injects a periodic runtime context after every N executed actions:

$$\mathcal{X}_n = \begin{cases} \mathcal{X}, & n \bmod N = 0, \\ \varnothing, & \text{otherwise}, \end{cases} \tag{17}$$

where n denotes the number of executed actions, N denotes the injection interval and is fixed as 5 for simplicity, and $\mathcal{X}$ denotes the runtime context that reinforces capability-aware invocation. Such periodic context helps preserve the capability-aware decision process throughout long-horizon execution. Thanks to the explicit capability reevaluation before action, CSI could enables retrieved skills to be invoked when their guidance becomes necessary, thereby alleviating invocation-side TSM.

3 Experiments

In this section, we conduct extensive experiments on three representative agent skill benchmarks to evaluate the effectiveness of SkillDreamer in both skill retrieval and end-to-end task execution. Due to space limitation, we present more experimental details and results in the Appendix.

3.1 Experimental Setup

Benchmarks and Metrics. We evaluate SkillDreamer on three representative agent skill benchmarks, including SkillRet (Cho et al., 2026), SkillUsage (Liu et al., 2026), and SkillNet-Gym (Liang et al., 2026). SkillRet focuses on large-scale skill retrieval with 4,997 test queries and 6,660 candidate skills, while SkillUsage and SkillNet-Gym further support executable end-to-end evaluation. SkillUsage contains 87 executable tasks and 34,396 candidate skills, of which 84 valid tasks are retained following its official protocol. Following their official protocols, we report Recall@K with $K \in \{5, 10, 15\}$ for skill retrieval, and *Pass* and *Load* for end-to-end task execution, measuring task completion and whether provided skills are loaded during execution, respectively. More benchmark statistics and evaluation details are provided in the Appendix.

Table 1: Retrieval results (%) on SkillUsage, SkillRet, and SkillNet-Gym. SkillDreamer is applied plug-and-play to representative retrievers. The best results within each backbone are shown in **bold**.

Retriever	Variant	SkillUsage		SkillRet		SkillNet-Gym	
		R@5	R@10	R@5	R@10	R@5	R@10
Sparse Retrieval							
BM25	Original	49.06	55.71	49.66	56.17	48.90	60.64
General Embedding Retrievers							
BGE-large-en-v1.5	Original	47.56	53.88	55.96	61.40	48.38	58.63
E5-large-v2	Original	46.42	48.29	51.51	57.42	43.11	51.01
Qwen3-Embedding-8B	Original	53.39	60.95	60.64	67.10	52.26	64.74
Qwen3-Embedding-0.6B	Original	52.68	62.25	59.14	64.91	48.60	61.77
	+ Ours	**60.42**	**69.47**	**66.93**	**71.73**	**60.92**	**69.51**
	Δ	+7.74	+7.22	+7.79	+6.82	+12.32	+7.74
Skill-oriented Retrievers							
SkillRouter-Embedding-0.6B	Original	46.55	52.29	71.01	76.00	39.62	48.56
	+ Ours	**56.15**	**61.69**	**75.54**	**81.00**	**61.36**	**68.43**
	Δ	+9.60	+9.40	+4.53	+5.00	+21.74	+19.87
SkillRet-Embedding-0.6B	Original	57.88	65.97	79.14	85.33	55.68	66.04
	+ Ours	**60.80**	**68.81**	**79.91**	**85.53**	**63.55**	**74.70**
	Δ	+2.92	+2.84	+0.77	+0.20	+7.87	+8.66
R3-Embedding	Original	55.50	61.17	81.82	87.64	54.70	66.79
	+ Ours	**59.98**	**67.51**	**82.76**	**87.81**	**66.39**	**75.59**
	Δ	+4.48	+6.34	+0.94	+0.17	+11.69	+8.80

Implementation Details. Unless otherwise specified, we employ Qwen3.6-27B (Qwen Team, 2026) for PSG and Qwen3-Embedding-0.6B (Zhang et al., 2025) as the default retriever. PSG infers at most five capability requirements for each task, while HSR retains $K_c = 20$ candidates for each inferred capability before reranking. For end-to-end evaluation, we provide the top-five retrieved skills as candidate skills to GPT-5.5 (OpenAI, 2026) and DeepSeek-V4-Flash (DeepSeek-AI, 2026). During execution, CSI is applied to determine whether a candidate skill should be invoked. Additional implementation details are provided in the Appendix.

3.2 Experiments on Skill Retrieval

To comprehensively evaluate SkillDreamer on skill retrieval, we compare it with eight representative retrievers from three categories, namely, i) sparse retrievers, including BM25 (Robertson & Zaragoza, 2009); ii) general embedding retrievers, including BGE-large-en-v1.5 (Xiao et al., 2024), E5-large-v2 (Wang et al., 2022), and Qwen3-Embedding at different scales (Zhang et al., 2025); and iii) skill-oriented retrievers, including SkillRouter (Zheng et al., 2026), SkillRet (Cho et al., 2026), and R3 (Wang et al., 2026). To examine whether prospective skill identification could mitigate TSM during retrieval across different retrieval paradigms, we further apply SkillDreamer to Qwen3-Embedding-0.6B and three skill-oriented retrievers in a plug-and-play manner.

From the results in Table 1, one could have the following observations and conclusions. i) **SkillDreamer consistently improves the retrieval performance across different benchmarks and retrievers.** For example, applying SkillDreamer to Qwen3-Embedding-0.6B improves R@5 by 7.74, 7.79, and 12.32 percentage points on SkillUsage, SkillRet, and SkillNet-Gym, respectively. Consistent improvements are also observed on three skill-oriented retrievers, demonstrating the effectiveness and broad compatibility of prospective skill identification in mitigating TSM during retrieval. ii) **Stronger semantic matching does not always lead to better identification of execution-useful skills.** Although SkillRouter is built upon Qwen3-Embedding and further trained with large-scale skill data, it underperforms the base model on both SkillUsage and SkillNet-Gym. In contrast, SkillDreamer substantially improves SkillRouter on both benchmarks, indicating that strengthening task–skill semantic matching alone cannot bridge the discrepancy between objective-oriented tasks and execution-oriented skills, and is therefore insufficient to resolve TSM during retrieval.

Table 2: End-to-end task execution results (%) on SkillUsage and SkillNet-Gym. *Pass* denotes the task completion rate, while *Load* denotes the proportion of execution trajectories in which the agent loads at least one provided skill. Best results among non-oracle settings are shown in **bold**.

Benchmark	SkillUsage				SkillNet-Gym			
Agent	GPT-5.5		DeepSeek-V4-Flash		GPT-5.5		DeepSeek-V4-Flash	
	Pass	Load	Pass	Load	Pass	Load	Pass	Load
Ground-Truth Skills	59.0	97.6	46.3	72.6	66.1	100.0	61.0	40.7
No Skills	46.8	–	27.5	–	69.5	–	61.0	–
Direct Retrieval	48.1	92.9	37.1	51.2	69.5	93.2	59.3	30.5
Ours	**50.6**	**96.4**	**41.4**	**77.4**	**72.9**	**96.6**	**66.1**	**47.5**

Table 3: Ablation studies of SkillDreamer for skill retrieval and invocation.

(a) Skill Retrieval

Setting	SkillRet		SkillUsage	
	R@5	R@10	R@5	R@10
Direct Retrieval	59.14	64.91	52.68	62.25
Pseudo Skill w/o Cap.	49.56	55.15	48.51	58.69
Pseudo-Skill Retrieval	55.56	62.60	48.85	59.00
PSG + HSR	**66.93**	**71.73**	**60.42**	**69.47**

(b) Skill Invocation

Setting	GPT-5.5		DeepSeek-V4-Flash	
	Pass	Load	Pass	Load
No Skills	46.8	–	27.5	–
Pseudo Skills	42.9	90.5	25.8	**78.6**
PSG + HSR	49.7	95.2	38.5	54.8
SkillDreamer	**50.6**	**96.4**	**41.4**	77.4

3.3 Experiments on End-to-End Task Execution

In this section, we further evaluate SkillDreamer on end-to-end task execution using GPT-5.5 and DeepSeek-V4-Flash. For comparison, we consider three settings: i) *Ground-Truth Skills* directly provides the annotated skills; ii) *No Skills* performs execution without external skill support; and iii) *Direct Retrieval* directly retrieves candidate skills using the original task. For all skill-assisted settings, the agent is provided with five candidate skills and autonomously decides whether to invoke the provided skills during execution. As shown in Table 2, even when ground-truth skills are directly provided, DeepSeek-V4-Flash achieves a Load of only 72.6% on SkillUsage, indicating that skill availability alone does not guarantee successful invocation. Compared with Direct Retrieval, the proposed SkillDreamer consistently improves both Pass and Load across different agents and benchmarks. These results demonstrate that prospective skill identification could facilitate effective skill identification and consequently improve downstream task execution.

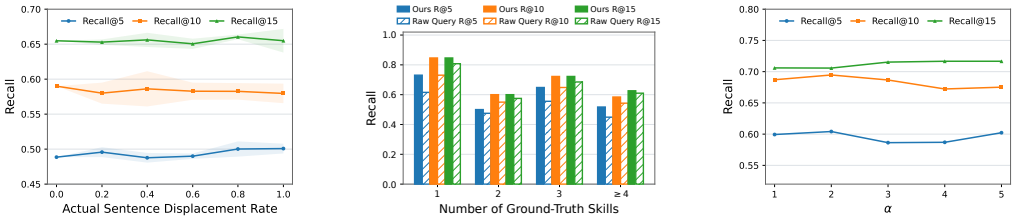
3.4 Ablation and Analytic Studies

In this section, we conduct ablation and analytic studies to further investigate the effectiveness of SkillDreamer. Unless otherwise specified, all experiments are conducted on SkillUsage with Qwen3-Embedding-0.6B as the retriever.

Ablation Study. To investigate the contribution of different designs in SkillDreamer, we construct several variants for the retrieval and invocation stages, respectively.

For skill retrieval, Table 3(a) compares: i) *Direct Retrieval*, which retrieves skills using the original task; ii) *Pseudo Skill w/o Cap.*, which directly generates pseudo skills from the task for retrieval; iii) *Pseudo-Skill Retrieval*, which first infers required capabilities through PSG and then retrieves with the generated pseudo skills; and iv) *PSG + HSR*, which denotes the complete retrieval stage. From the results, PSG + HSR achieves the best performance, demonstrating that explicitly reasoning about required capabilities could better bridge task objectives and execution-oriented skills for skill identification.

For skill invocation, Table 3(b) compares: i) *No Skills*, which performs the task without external skills; ii) *Pseudo Skills*, which directly provides the generated pseudo skills for execution; iii) *PSG + HSR*, which performs execution with the skills retrieved by our method; and iv) *SkillDreamer*, which further employs CSI for skill identification during execution. Introducing CSI upon PSG



(a) Pseudo-skill robustness. (b) Number of ground-truth skills. (c) Sensitivity to α.

Figure 3: Analytic studies of SkillDreamer. (a) Retrieval robustness under pseudo-skill sentence displacement. (b) Recall@K under different numbers of ground-truth skills. (c) Sensitivity analysis of the hyperparameter α on SkillUsage.

+ HSR consistently improves both Pass and Load, demonstrating that reasoning about capability requirements during execution could facilitate the identification and invocation of desired skills, thereby mitigating TSM at the invocation stage.

Robustness to Pseudo Skill Generation. To examine how the quality of generated pseudo skills affects SkillDreamer, we randomly reorder sentences within each pseudo skill at varying displacement rates under the “Pseudo-Skill Retrieval" setting. As shown in Fig. 3a, retrieval performance remains stable even under substantial sentence displacement. This result suggests that an approximate description of how the required capabilities could be realized is sufficient to bridge task objectives and execution-oriented skills. In other words, SkillDreamer does not require generated pseudo skills to preserve precise procedural knowledge.

Analysis on the Number of Ground-Truth Skills. We group SkillUsage tasks by the number of ground-truth skills to examine retrieval performance under varying capability requirements. As shown in Fig. 3b, SkillDreamer consistently outperforms Direct Retrieval across all groups, with clear gains retained as the number of required skills increases. These results demonstrate that SkillDreamer could effectively identify the diverse skills required by tasks with multiple capability requirements.

Parameter Analysis. In HSR, α balances the contribution of original-task relevance and prospective execution information during retrieval. To investigate its sensitivity, we vary α from 1 to 5 on SkillUsage. As shown in Fig. 3c, SkillDreamer maintains stable performance across different values of α, while $\alpha = 2$ achieves competitive results on all metrics. This observation indicates that HSR does not rely on a carefully tuned balance between task relevance and prospective execution information.

4 Conclusion

In this work, we reveal and study the Task–Skill Misalignment (TSM) problem in agent skill identification, where task descriptions mainly specify task objectives while skill documents provide the capabilities and procedural guidance required for execution. We empirically demonstrate that such a discrepancy could hinder desirable skills from being correctly identified during both skill retrieval and invocation. To address this problem, inspired by human prospective cognition, we propose SkillDreamer, a prospective skill identification framework that explicitly reasons about the capabilities required for execution to bridge task objectives and execution-oriented skills. Extensive experiments on representative benchmarks not only demonstrate the necessity of mitigating TSM, but also verify the effectiveness and generalizability of SkillDreamer across diverse retrievers and LLM agents. In the future, we plan to extend SkillDreamer to broader settings, including coding and GUI agents, and investigate how agents can more reliably follow the procedural guidance provided by identified skills.

AI USE STATEMENT

In this work, we used generative AI tools to assist with the analysis of experimental results and to improve the clarity, grammar, and academic presentation of the manuscript. Beyond the LLM-based methods and evaluation procedures described in this paper, generative AI was not used to generate experimental data. All AI-assisted analyses and edits were carefully reviewed, verified, and revised by the authors as needed. The authors made the final interpretations and conclusions and take full responsibility for the accuracy and integrity of the manuscript.

REFERENCES

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

Chishui Chen, Jiaye Lin, Te Sun, Yi Yang, Junxi Wang, Cong Qin, Yangen Hu, Lu Pan, and Ke Zeng. Skill or skip? learning selective skill invocation in agentic tasks via dual-granularity preference learning. *arXiv preprint arXiv:2606.00510*, 2026a.

Tianyi Chen, Yinheng Li, Michael Solodko, Sen Wang, Nan Jiang, Tingyuan Cui, Junheng Hao, Jongwoo Ko, Sara Abdali, Leon Xu, et al. Cua-skill: Develop skills for computer using agent. *arXiv preprint arXiv:2601.21123*, 2026b.

Yanping Chen, Weijie Shi, Wen Yang, and Jiajie Xu. Task decomposition-guided reranking for adaptive agent skill retrieval. *arXiv preprint arXiv:2607.06283*, 2026c.

Hongcheol Cho, Ryangkyung Kang, and Youngeun Kim. Skillret: A large-scale benchmark for skill retrieval in llm agents. *arXiv preprint arXiv:2605.05726*, 2026.

DeepSeek-AI. DeepSeek-V4: Towards highly efficient million-token context modeling. *arXiv preprint arXiv:2606.19348*, 2026.

Xueping Gao. Compositional skill routing for llm agents: Decompose, retrieve, and compose. *arXiv preprint arXiv:2606.18051*, 2026.

Daniel T Gilbert and Timothy D Wilson. Prospection: Experiencing the future. *Science*, 317(5843): 1351–1354, 2007.

Jiaxi Li, Ke Deng, Yun Wang, Jingyuan Huang, Yucheng Shi, Qiaoyu Tan, Jin Lu, and Ninghao Liu. Online skill learning for web agents via state-grounded dynamic retrieval. *arXiv preprint arXiv:2606.04391*, 2026.

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

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. *arXiv preprint arXiv:2604.04323*, 2026.

OpenAI. Introducing GPT-5.5, April 2026. URL https://openai.com/index/introducing-gpt-5-5/.

Qwen Team. Qwen3.6-27B: Flagship-level coding in a 27b dense model, April 2026. URL https://qwen.ai/blog?id=qwen3.6-27b.

Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Information Retrieval*, 3(4):333–389, 2009.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551, 2023.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*, 2022.

Zifei Wang, Wei Wen, Qiang Ji, Ruizhi Qiao, and Xing Sun. Skill is not document: A query-conditional benchmark and two-stage retriever for llm agent skill routing. *arXiv preprint arXiv:2606.03565*, 2026.

Zora Zhiruo Wang, Apurva Gandhi, Graham Neubig, and Daniel Fried. Inducing programmatic skills for agentic tasks. *arXiv preprint arXiv:2504.06821*, 2025.

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.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, et al. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025.

YanZhao Zheng, ZhenTao Zhang, Chao Ma, YuanQiang Yu, JiHuai Zhu, Yong Wu, Tianze Xu, Baohua Dong, Hangcheng Zhu, Ruohui Huang, et al. Skillrouter: Skill routing for llm agents at scale. *arXiv preprint arXiv:2603.22455*, 2026.

CONTENTS

A Introduction **13**

B Related Work **13**

B.1 Information Retrieval 13

B.2 Agent Skill Retrieval 13

B.3 Agent Skill Invocation 13

C Additional Details of Implementation **14**

C.1 Details of the Algorithm 14

C.2 Details of the Datasets 16

C.3 Details of the Prompts 16

C.4 Details of the Baselines 18

C.5 Details of the Experimental Settings 19

C.6 Details of the Evaluation Metrics 19

D Additional Experiments **20**

D.1 Unified Prospective Retrieval and Execution 20

D.2 Additional Retrieval Results 20

D.3 Generalization to SRA-Bench 22

D.4 Generalization across Diverse Backbones 22

D.5 Additional Ablation Results 24

D.6 Additional Analysis on the Number of Skills 24

References **25**

A Introduction

In this supplementary material, we provide further implementation details and additional experimental results to demonstrate the effectiveness of SkillDreamer.

B Related Work

In this section, we provide a brief review of three topics highly related to this work, including information retrieval, agent skill retrieval, and agent skill invocation.

B.1 Information Retrieval

Information retrieval aims to identify relevant information from large-scale corpora for a given query. Existing approaches mainly estimate query–document relevance through lexical matching or learned semantic representations (Robertson & Zaragoza, 2009; Karpukhin et al., 2020). With the emergence of LLMs, recent studies further improve retrieval by enriching query-side information or reasoning over candidate relevance. Based on how LLMs are incorporated into the retrieval process, existing approaches could be roughly divided into two categories: i) pre-retrieval query enhancement methods (Ma et al., 2023; Wang et al., 2023; Gao et al., 2023), which reformulate, expand, or decompose the original query, or synthesize auxiliary retrieval content to better express the underlying information need; ii) post-retrieval relevance refinement methods (Sun et al., 2023; Qin et al., 2024; Reddy et al., 2024), which employ LLMs to reason over retrieved candidates and rerank them through pairwise or listwise comparisons.

Different from conventional information retrieval, agent skill identification is not merely concerned with finding information relevant to a query, but with identifying reusable capabilities and procedural guidance that could support task execution. Although existing query enhancement methods enrich the information expressed by the original query, they primarily aim to better characterize the underlying information need without explicitly reasoning about the capabilities required for execution. In contrast, SkillDreamer prospectively infers execution-required capabilities and imagines their realizations as skill-like procedural guidance, thereby bridging task objectives and execution-oriented skills. Moreover, SkillDreamer extends such capability-aware identification beyond retrieval to skill invocation during execution.

B.2 Agent Skill Retrieval

Agent skill retrieval aims to identify useful skills from a large-scale skill gallery for a given task, thereby providing reusable procedural guidance for subsequent execution. According to how task–skill association is improved, existing studies could be roughly divided into two categories: i) model adaptation methods, which adapt general-purpose retrieval models to the skill domain using paired task descriptions and skills (Cho et al., 2026; Wang et al., 2026); ii) pipeline redesign methods, which either adopt multi-stage retrieval and reranking strategies (Zheng et al., 2026; Li et al., 2026a), or exploit task and skill structures through query decomposition, dependency modeling, and structured organization (Chen et al., 2026c; Zeng et al., 2026; Ding, 2026).

Different from existing studies that mainly improve the direct association between task descriptions and skill documents, our work revisits skill retrieval from the perspective of task execution. Task descriptions mainly specify task objectives, whereas skill documents describe the capabilities and procedural guidance required for execution, giving rise to Task–Skill Misalignment (TSM) during retrieval. As a result, execution-critical skills might be under-prioritized when their semantics are weakly aligned with the original task description. To mitigate such misalignment, SkillDreamer explicitly reasons about the capabilities required for task execution and imagines how they could be realized as pseudo skills, thereby introducing prospective execution information to bridge task objectives and execution-oriented skills.

B.3 Agent Skill Invocation

Agent skill invocation aims to determine whether and which available skills should be invoked during task execution, thereby providing appropriate procedural guidance when needed. Based on how

invocation decisions are made, existing approaches could be roughly divided into two categories: i) invocation policy optimization methods, which explicitly optimize whether or which skill to invoke through dedicated credit assignment or preference learning (Li et al., 2026b; Chen et al., 2026a); ii) execution-conditioned invocation methods, which adapt skill invocation to evolving execution states using explicit interaction trajectories or latent agent representations (Wu et al., 2026; Chen et al., 2026b).

Different from existing studies that directly derive skill invocation from the task or execution context, our work explicitly models the capabilities required by emerging subtasks. During execution, emerging subtasks primarily specify what needs to be accomplished, whereas available skills describe the capabilities and procedural guidance they provide, leaving the capability requirements connecting the two implicit. Such misalignment might cause desirable skills to remain uninvoked even when they have already been retrieved. To address this issue, SkillDreamer continually reassesses the capabilities required by emerging subtasks and accordingly determines whether and which retrieved skills should be invoked.

C Additional Details of Implementation

In this section, we provide additional implementation details of SkillDreamer. Specifically, we first present the prospective skill retrieval procedure in the form of pseudocode. We then introduce the details of the datasets, prompts, and baseline retrievers used in our experiments. Finally, we present the experimental settings and evaluation metrics to facilitate reproducibility.

C.1 Details of the Algorithm

To present the retrieval procedure clearly, we provide the algorithmic details of PSG and HSR in Algorithm 1. The notation follows the method section of the main paper.

Algorithm 1 Prospective skill retrieval in SkillDreamer.

Input: Task q, skill gallery $\mathcal{S} = \{s_i\}_{i=1}^{G}$, LLM $\mathcal{M}$, embedding model f, retrieval scales K_c, K_r, and K, fusion weight α
Output: Retrieved candidate skill set $\mathcal{C}$

Infer capability requirements and semantic anchors: $(\mathcal{U}, \mathcal{A}) \leftarrow \mathcal{M}(q; \mathcal{P}_{\mathrm{PSG}}^{\mathrm{inf}})$
Let $M = |\mathcal{U}|$
for each capability $u_j \in \mathcal{U}$ **do**
    Generate the corresponding pseudo skill: $\tilde{s}_j \leftarrow \mathcal{M}(u_j, \mathcal{A}; \mathcal{P}_{\mathrm{PSG}}^{\mathrm{gen}})$
end for
Let $\widetilde{\mathcal{S}} = \{\tilde{s}_j\}_{j=1}^{M}$
Encode the task, pseudo skills, and gallery skills: $\mathbf{z}_q = f(q)$, $\tilde{\mathbf{z}}_j = f(\tilde{s}_j)$, and $\mathbf{z}_i = f(s_i)$
Construct the lexical query a by concatenating the anchors in $\mathcal{A}$
for each pseudo skill $\tilde{s}_j \in \widetilde{\mathcal{S}}$ **do**
    for each gallery skill $s_i \in \mathcal{S}$ **do**
        Compute query–skill relevance: $r_i^q = \mathrm{sim}(\mathbf{z}_q, \mathbf{z}_i)$
        Compute pseudo-skill–skill relevance: $r_{ij}^p = \mathrm{sim}(\tilde{\mathbf{z}}_j, \mathbf{z}_i)$
        Compute hybrid retrieval score: $d_{ij} = \alpha r_i^q + r_{ij}^p$
    end for
    Retain the top-K_c skills ranked by d_{ij} as the preliminary candidate set $\mathcal{B}_j$
    for each candidate skill $s_i \in \mathcal{B}_j$ **do**
        Compute anchor relevance: $r_i^a = \mathrm{BM25}(a, s_i)$
    end for
    Apply min–max normalization to the three relevance scores in the current candidate set to obtain $\bar{r}_i^q$, $\bar{r}_{ij}^p$, and $\bar{r}_i^a$
    for each candidate skill $s_i \in \mathcal{B}_j$ **do**
        Compute the final reranking score: $h_{ij} = \alpha \bar{r}_i^q + \bar{r}_{ij}^p + \bar{r}_i^a$
    end for
    Rank $\mathcal{B}_j$ by h_{ij} to obtain $\mathcal{L}_j$ and retain its top-K_r skills
end for
Let $\pi_j(v)$ be the index of the v-th ranked skill in $\mathcal{L}_j$
Construct the ranking matrix: $\mathbf{R} \leftarrow \begin{bmatrix} s_{\pi_1(1)} & \cdots & s_{\pi_M(1)} \\ \vdots & \ddots & \vdots \\ s_{\pi_1(K_r)} & \cdots & s_{\pi_M(K_r)} \end{bmatrix}$
Initialize $\mathcal{C} \leftarrow \varnothing$
Scan $\mathbf{R}$ row by row from left to right, appending each unseen skill to $\mathcal{C}$ until $|\mathcal{C}| = K$
return $\mathcal{C}$

C.2 Details of the Datasets

We evaluate the proposed method SkillDreamer on three skill-retrieval datasets: SkillRet (Cho et al., 2026), SkillUsage (Liu et al., 2026), and SkillNet-Gym (Liang et al., 2026).

- SkillRet is a large-scale benchmark constructed from public skills collected from open-source communities. It generates natural language queries from sampled skills using an LLM, with the sampled skills serving as the corresponding retrieval targets.
- SkillUsage is a benchmark built upon SkillsBench (Li et al., 2026c). It contains executable tasks and their corresponding skills, which are integrated into a large-scale library of real-world skills for skill retrieval evaluation.
- SkillNet-Gym is the execution benchmark introduced by SkillNet (Liang et al., 2026). We use its tasks and ground-truth skills but construct a different retrieval gallery by merging them with the SkillUsage gallery and deduplicating the normalized `SKILL.md` content.

Detailed statistics and the numbers of tasks used in the retrieval and end-to-end execution evaluations are summarized in Table 4.

Table 4: Statistics of the datasets used in our experiments. The SkillNet-Gym gallery combines its 162 unique ground-truth skills with the SkillUsage gallery and contains 34,522 skills after normalization and deduplication.

Dataset	Split	Retrieval Tasks	End-to-End Tasks	Skills	GT Skills
SkillRet	Test	4,997	–	6,660	1–3
SkillUsage	Test	87	84	$\sim$34K	Varies
SkillNet-Gym	Test	159	59	34,522	Varies

As shown in Table 4, the retrieval evaluation on SkillRet uses its complete test split of 4,997 tasks. For SkillUsage, retrieval is evaluated on all 87 tasks, while end-to-end execution is evaluated on 84 tasks. For SkillNet-Gym, retrieval is evaluated on all 159 tasks, while end-to-end execution uses the first 59 tasks in the benchmark ordering. For both SkillUsage and SkillNet-Gym, we use the same two agent configurations in the end-to-end evaluation: GPT-5.5 with Codex and DeepSeek-V4-Flash with QCoder. All results reported in the main paper are computed over their corresponding evaluation subsets.

C.3 Details of the Prompts

To facilitate reproducibility, we provide the prompts used by the Prospective Skill Generation (PSG) and Capability-aware Skill Invocation (CSI) modules described in the main paper. Table 5 details the capability-inference prompt $\mathcal{P}^{\text{inf}}_{\text{PSG}}$, which jointly infers independent capability requirements $\mathcal{U}$ and task-specific semantic anchors $\mathcal{A}$ from the original task. Table 6 shows the prospective-generation prompt $\mathcal{P}^{\text{gen}}_{\text{PSG}}$, which takes each inferred capability and the semantic anchors as input and generates a corresponding skill-like pseudo skill with concrete procedural guidance. Table 7 provides the capability-aware invocation instruction $\mathcal{P}_{\text{CSI}}$, which infers the capability required by the current objective and determines whether an appropriate candidate skill should be invoked. Table 8 gives the periodic runtime checkpoint $\mathcal{X}_n$, which is injected after every fixed number of execution steps to trigger another capability-gap check while accounting for the remaining execution budget.

Table 5: Capability-inference prompt $\mathcal{P}_{\text{PSG}}^{\text{inf}}$ used by the PSG module.

Role. You are an expert in decomposing tasks into capability requirements for skill retrieval.
Raw Task. `{query}`
Goal. Infer the independent capabilities required to complete the raw task and extract task-specific semantic anchors, including relevant domain terminology, key objects, and task targets. Each capability will later be transformed into a skill-like retrieval proxy.

Decomposition Principles.

- Decompose the task by independent capability requirements, rather than execution steps.
- Preserve relevant technical domains, methods, tools, and constraints from the raw task.
- Specialized artifact processing may be treated as an independent capability when it requires specific technical expertise.
- Do not invent capabilities, tools, methods, or requirements unsupported by the raw task.

Output Requirements.
Return JSON only. Generate one to five capability requirements and a concise list of semantic anchors grounded in the raw task. Each capability should be independent, self-contained, and useful for specialized skill retrieval.
Output Format.

```
{
  "capabilities": [
    {
      "id": "u1",
      "name": "Capability name",
      "requirement": "Self-contained capability requirement".
    }
  ],
  "semantic_anchors": [
    "Domain term, key object, or task target from the raw task".
  ]
}
```

Table 6: Prospective-generation prompt $\mathcal{P}_{\text{PSG}}^{\text{gen}}$ used by the PSG module.

Role. Write one concise, retrieval-oriented skill for a single capability.
Inputs. `{capability_requirement}` and `{semantic_anchors}`
Goal. Generate one hypothetical `SKILL.md` that describes how the inferred capability could be realized through concrete procedural guidance. Preserve the relevant semantic anchors so that the pseudo skill remains specific to the original task context.

Generation Principles.

- Teach how to realize the capability instead of merely restating its description.
- Incorporate concrete domain knowledge, such as methods, algorithms, APIs, workflows, failure handling, or validation.
- Preserve capability-defining technical terms and constraints.
- Do not invent unsupported facts or broaden the skill into adjacent capabilities.
- Produce a concise, reusable, and standalone skill using only capability-specific sections.

Output Requirements.
Return only the complete `SKILL.md` content. The generated skill must follow the specified metadata structure and contain a concise skill name, trigger-oriented description, and a human-readable title.
Output Format.

```
--
name: short-lowercase-kebab-case-name
description: "Concise, trigger-oriented description of the
capability."
--
# Human-Readable Skill Title
Write the task-adaptive skill body.
```

Table 7: Capability-aware invocation instruction $\mathcal{P}_{\text{CSI}}$ used by the CSI module.

Capability-aware Skill Invocation. Complete the task with the available tools and the fixed set of candidate skills exposed by the runtime. Do not search for, install, or create additional skills.
Inputs. Use the sub-task sequence formed so far, the metadata of all retrieved candidate skills, and the runtime context supplied at the current checkpoint.
Goal. Before the next action, reassess the current objective, infer the capability required to accomplish it, and determine whether an appropriate candidate skill should be invoked. If no candidate skill provides the required capability, proceed without additional skill guidance.
When to Check. Run a check whenever the runtime issues a `RUNTIME CAPABILITY CHECKPOINT`. Also check when the required capability may have changed, the next step is unclear, an approach fails, attempts repeat, or new evidence makes the current skill match uncertain.
Check and Match. Before any tool call, begin the response with exactly five concise lines:

```
SUBGOAL: the immediate subgoal
EVIDENCE: relevant evidence and unresolved constraints
CAPABILITY: the capability needed to resolve the most important
unresolved issue
DECISION: MATCH, PARTIAL_MATCH, NO_MATCH, or NO_CAPABILITY_GAP; include
the selected skill when applicable
NEXT: one bounded action and its observable success check
```

Infer `CAPABILITY` from the unresolved objective, requirement, failure, or constraint, not from the tool action or method already planned. Compare it with all visible candidate-skill metadata before choosing `NO_CAPABILITY_GAP`.

- Use `MATCH` when a skill covers the required capability, `PARTIAL_MATCH` when it covers a necessary part, and `NO_MATCH` when none does.
- Prefer the most specific skill that addresses the unresolved issue. Do not select a skill merely because it is the closest candidate.
- Use `NO_CAPABILITY_GAP` only when the available evidence supports the current method, it satisfies the known constraints, and no unresolved failure or issue could invalidate `NEXT`.

Act and Verify. For `MATCH` or `PARTIAL_MATCH`, load the selected skill before any other tool action. Follow its applicable procedure and use its available implementation unless observed evidence shows that it is unsuitable. Load a different skill only when new evidence changes the match.
For `NO_CAPABILITY_GAP`, perform `NEXT`. For `NO_MATCH`, decompose the subgoal or make one small, materially different attempt with the permitted tools; continue independent work and report any remaining blocker.
Execute `NEXT` and verify its stated success check. If it fails or conflicts with known evidence, run another check before continuing. Do not repeat a failed approach without new evidence or a different method. Before completion, inspect the persisted result itself and validate its structure, content, and required outputs against the task's observable requirements.

Table 8: Periodic runtime capability-checkpoint prompt $\mathcal{X}_n$ used to trigger the CSI module.

```
RUNTIME CAPABILITY CHECKPOINT (periodic): {completedSteps} execution
steps have completed. Agent execution budget: {budgetSeconds}
seconds; elapsed: {elapsedSeconds} seconds; estimated remaining:
{remainingSeconds} seconds. Before taking another action, apply the
Runtime Capability-Gap Recovery protocol in the task instructions.
Infer the capability needed for the immediate subgoal and compare it
with the visible skill metadata. Record NO_CAPABILITY_GAP, MATCH,
PARTIAL_MATCH, or NO_MATCH, then act on the result. Choose a NEXT
action that can be completed and verified within the remaining budget
while leaving time to recover from failure.
```

C.4 Details of the Baselines

We consider three categories of retrieval baselines in our experiments. The details of the corresponding baseline retrievers are provided as follows:

- Sparse Retrieval. BM25 (Robertson & Zaragoza, 2009) is adopted as the representative sparse retriever, ranking skills through term-level matching between the query and skill documents.

- General Embedding Retrieval. Qwen3-Embedding (Zhang et al., 2025) performs instruction-aware dense semantic matching, while BGE-large-en-v1.5 (Xiao et al., 2024) combines RetroMAE pretraining with large-scale contrastive learning, and E5-large-v2 (Wang et al., 2022) learns general-purpose representations from weakly supervised text pairs followed by InfoNCE-based contrastive training.
- Skill-oriented Retrieval. SkillRouter-Embedding-0.6B (Zheng et al., 2026), SkillRet-Embedding-0.6B (Cho et al., 2026), and R3-Embedding (Wang et al., 2026) are all optimized with LLM-generated query–skill supervision, but differ in their training strategies. SkillRouter adopts in-batch InfoNCE with semantic, lexical, and category-based hard negatives, whereas SkillRet uses MultipleNegativesRankingLoss with in-batch negatives. Beyond pairwise relevance, R3-Embedding is trained on queries associated with one or more relevant skills and adopts multi-positive InfoNCE with a sibling-reward term to preserve multi-skill relevance.

C.5 Details of the Experimental Settings

To facilitate reproducibility, we provide the detailed experimental settings of SkillDreamer. As shown in Algorithm 1, for each inferred capability u_j, K_c denotes the number of skills retained in the preliminary candidate set $\mathcal{B}_j$ according to the hybrid score d_{ij} (Line 15), while K_r denotes the number of skills retained from the reranked list $\mathcal{L}_j$ (Line 23). The resulting capability-specific rankings are merged in round-robin order with duplicates removed until K unique skills are collected in the final candidate set $\mathcal{C}$ (Line 28). Unless otherwise specified, we set $K_c = 20$, $K_r = 20$, and $K = 15$.

Beyond the retrieval scales, we further specify the fusion weights for different embedding backbones. Algorithm 1 follows the main-paper formulation, $d_{ij} = \alpha r_i^q + r_{ij}^p$ and $h_{ij} = \alpha \bar{r}_i^q + \bar{r}_{ij}^p + \bar{r}_i^a$. To state the model-specific implementation in a unified form, we write the same scores as

$$d_{ij} = \alpha r_i^q + \beta r_{ij}^p, \tag{18}$$

$$h_{ij} = \alpha \bar{r}_i^q + \beta \bar{r}_{ij}^p + \gamma \bar{r}_i^a. \tag{19}$$

The main-paper formulation is recovered when $\beta = \gamma = 1$. By default, we set $\alpha = 2$, $\beta = 1$, and $\gamma = 1$. For SkillRet-Embedding-0.6B and R3-Embedding on the SkillRet benchmark, we use $\alpha = 4$, $\beta = 1$, and $\gamma = 0.25$. Notably, this alternative configuration is used exclusively to evaluate the applicability of SkillDreamer to SkillRet-Embedding-0.6B and R3-Embedding on SkillRet, while all other experiments retain the fixed default configuration.

C.6 Details of the Evaluation Metrics

We evaluate SkillDreamer from two complementary perspectives: skill retrieval and end-to-end task execution. For each query q, let G_q denote its ground-truth skill set and $\mathcal{C}_q$ denote the final candidate set containing the top-K retrieved skills. Using these definitions, we evaluate retrieval performance with the following three metrics:

- Recall@K measures the proportion of ground-truth skills retrieved within the top-K results:
$$\mathrm{Recall}_q@K = \frac{|\mathcal{C}_q \cap G_q|}{|G_q|}. \tag{20}$$
- NDCG@K further considers the ranking positions of retrieved ground-truth skills, assigning larger gains to relevant skills appearing at higher ranks:
$$\mathrm{NDCG}_q@K = \frac{\sum_{i=1}^{K} \frac{\mathrm{rel}_i}{\log_2(i+1)}}{\sum_{i=1}^{\min(|G_q|,K)} \frac{1}{\log_2(i+1)}}, \tag{21}$$
where $\mathrm{rel}_i = 1$ if the skill at rank i belongs to G_q, and 0 otherwise.
- Completeness@K is a stricter metric that evaluates whether all ground-truth skills are retrieved within the top-K results:
$$\mathrm{Completeness}_q@K = \mathbb{I}[G_q \subseteq \mathcal{C}_q]\,. \tag{22}$$

We further evaluate end-to-end task execution using the following metrics:

- Pass Rate measures task-execution performance by averaging the verifier rewards over all tasks:

$$\text{PassRate} = \frac{1}{N}\sum_{i=1}^{N} r_i, \tag{23}$$

  where r_i denotes the verifier reward for task i, with missing or invalid rewards treated as zero.

- Load Rate measures the proportion of tasks for which the agent loads at least one provided skill during execution:

$$\text{LoadRate} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}[|U_i| > 0], \tag{24}$$

  where U_i denotes the set of provided skills loaded by the agent for task i.

D Additional Experiments

In this section, we present additional experimental results to further demonstrate the effectiveness of SkillDreamer. Specifically, we first provide additional end-to-end execution results to examine the impact of improved retrieval on downstream task performance. We then report complementary retrieval results and evaluate its generalization across SRA-Bench (Su et al., 2026) and diverse retriever backbones. Finally, we provide additional ablation results and further investigate the effect of the number of ground-truth skills on SkillRet.

D.1 Unified Prospective Retrieval and Execution

We further investigate a unified setting in which Qwen3.6-27B performs both prospective capability reasoning and task execution using its default QCoder harness. We compare SkillDreamer with *Raw Query Retrieval*, which retrieves skills directly from the original task query. As shown in Table 9, SkillDreamer improves the Pass rate from 31.3% to 37.6% (+6.3 points), while the Load rate remains unchanged at 59.5%. The gain without more frequent skill loading indicates that prospective reasoning retrieves complementary skills that better support task completion rather than merely increasing skill usage.

Table 9: End-to-end execution results (%) on SkillUsage using Qwen3.6-27B in the QCoder harness. Best values are in **bold**.

Skill Setting	Pass	Load
Raw Query Retrieval	31.3	**59.5**
SkillDreamer	**37.6**	**59.5**
Δ	+6.3	+0.0

D.2 Additional Retrieval Results

Tables 10, 11, and 12 report complete Recall@K, NDCG@K, and Completeness@K results on SkillUsage, SkillRet, and SkillNet-Gym. With Qwen3-Embedding-0.6B, SkillDreamer improves R@5 by 7.74, 7.79, and 12.32 points; NDCG@5 by 4.79, 4.68, and 10.68 points; and Completeness@5 by 6.90, 9.76, and 11.95 points, respectively, demonstrating consistent gains in both ranking quality and complete skill-set recovery.

Table 10: Complete SkillUsage retrieval results (%); per-backbone best results are in **bold**.

Retriever	Setting	Recall			NDCG			Comp.		
		@5	@10	@15	@5	@10	@15	@5	@10	@15
Sparse Retrieval										
BM25	Original	49.06	55.71	61.59	48.34	50.86	52.79	33.33	40.23	45.98
General Embedding Retrievers										
BGE-large-en-v1.5	Original	47.56	53.88	56.84	48.24	50.92	51.98	32.18	35.63	39.08
E5-large-v2	Original	46.42	48.29	50.48	46.26	46.93	47.74	32.18	32.18	33.33
Qwen3-Embedding-8B	Original	53.39	60.95	69.72	54.77	57.91	60.78	32.18	41.38	50.57
	Original	52.68	62.25	67.66	52.62	56.51	58.36	35.63	43.68	48.28
Qwen3-Embedding-0.6B	**+ Ours**	**60.42**	**69.47**	**70.57**	**57.41**	**60.83**	**61.33**	**42.53**	**51.72**	**52.87**
	Δ	+7.74	+7.22	+2.91	+4.79	+4.32	+2.97	+6.90	+8.04	+4.59
Skill-oriented Retrievers										
	Original	46.55	52.29	55.70	47.32	49.78	51.00	29.89	34.48	35.63
SkillRouter-Embedding-0.6B	**+ Ours**	**56.15**	**61.69**	**65.21**	**55.57**	**57.80**	**58.97**	**33.33**	**42.53**	**45.98**
	Δ	+9.60	+9.40	+9.51	+8.25	+8.02	+7.97	+3.44	+8.05	+10.35
	Original	57.88	65.97	69.49	57.08	60.42	61.56	40.23	50.57	55.17
SkillRet-Embedding-0.6B	**+ Ours**	**60.80**	**68.81**	**74.92**	**59.39**	**62.59**	**64.55**	**42.53**	**54.02**	**58.62**
	Δ	+2.92	+2.84	+5.43	+2.31	+2.17	+2.99	+2.30	+3.45	+3.45
	Original	55.50	61.17	68.74	55.94	58.47	60.96	37.93	43.68	52.87
R3-Embedding	**+ Ours**	**59.98**	**67.51**	**71.43**	**58.16**	**61.37**	**62.56**	**40.23**	**51.72**	**55.17**
	Δ	+4.48	+6.34	+2.69	+2.22	+2.90	+1.60	+2.30	+8.04	+2.30

Table 11: Complete SkillRet retrieval results (%); per-backbone best results are in **bold**.

Retriever	Setting	Recall			NDCG			Comp.		
		@5	@10	@15	@5	@10	@15	@5	@10	@15
Sparse Retrieval										
BM25	Original	49.66	56.17	59.71	46.06	48.47	49.54	34.48	40.70	44.19
General Embedding Retrievers										
BGE-large-en-v1.5	Original	55.96	61.40	64.31	53.75	55.81	56.71	39.30	44.63	47.49
E5-large-v2	Original	51.51	57.42	60.86	48.01	50.21	51.28	36.54	41.95	45.17
Qwen3-Embedding-8B	Original	60.64	67.10	70.39	57.52	59.98	61.01	43.33	49.99	53.53
	Original	59.14	64.91	68.20	56.21	58.40	59.42	41.83	47.21	50.47
Qwen3-Embedding-0.6B	**+ Ours**	**66.93**	**71.73**	**74.48**	**60.89**	**62.69**	**63.52**	**51.59**	**56.95**	**60.16**
	Δ	+7.79	+6.82	+6.28	+4.68	+4.29	+4.10	+9.76	+9.74	+9.69
Skill-oriented Retrievers										
	Original	71.01	76.00	78.71	68.88	70.82	71.67	53.47	59.32	62.60
SkillRouter-Embedding-0.6B	**+ Ours**	**75.54**	**81.00**	**83.49**	**70.52**	**72.56**	**73.32**	**61.50**	**68.44**	**71.72**
	Δ	+4.53	+5.00	+4.78	+1.64	+1.74	+1.65	+8.03	+9.12	+9.12
	Original	79.14	85.33	88.01	75.58	**78.01**	**78.84**	66.02	74.98	78.89
SkillRet-Embedding-0.6B	**+ Ours**	**79.91**	**85.53**	**88.36**	**75.70**	77.87	78.75	**67.54**	**75.71**	**79.77**
	Δ	+0.77	+0.20	+0.35	+0.12	−0.14	−0.09	+1.52	+0.73	+0.88
	Original	81.82	87.64	**90.20**	78.79	**81.06**	**81.89**	68.78	77.43	81.33
R3-Embedding	**+ Ours**	**82.76**	**87.81**	90.19	**78.85**	80.83	81.57	**70.70**	**78.31**	**81.95**
	Δ	+0.94	+0.17	−0.01	+0.06	−0.23	−0.32	+1.92	+0.88	+0.62

Table 12: Complete SkillNet-Gym retrieval results (%); per-backbone best results are in **bold**.

Retriever	Setting	Recall			NDCG			Comp.		
		@5	@10	@15	@5	@10	@15	@5	@10	@15
Sparse Retrieval										
BM25	Original	48.90	60.64	64.26	45.17	50.70	52.02	22.64	42.77	46.54
General Embedding Retrievers										
BGE-large-en-v1.5	Original	48.38	58.63	60.82	47.34	52.30	53.17	18.87	30.19	32.70
E5-large-v2	Original	43.11	51.01	55.20	41.61	45.44	47.07	14.47	24.53	29.56
Qwen3-Embedding-8B	Original	52.26	64.74	67.59	50.29	56.08	57.15	22.64	45.91	49.69
Qwen3-Embedding-0.6B	Original	48.60	61.77	67.58	48.14	54.35	56.52	22.01	38.99	47.17
	+ Ours	**60.92**	**69.51**	**71.56**	**58.82**	**62.96**	**63.71**	**33.96**	**47.17**	**50.31**
	Δ	+12.32	+7.74	+3.98	+10.68	+8.61	+7.19	+11.95	+8.18	+3.14
Skill-oriented Retrievers										
SkillRouter-Embedding-0.6B	Original	39.62	48.56	53.14	39.08	43.36	45.18	16.98	23.27	26.42
	+ Ours	**61.36**	**68.43**	**70.59**	**59.40**	**62.74**	**63.56**	**34.59**	**48.43**	**49.69**
	Δ	+21.74	+19.87	+17.45	+20.32	+19.38	+18.38	+17.61	+25.16	+23.27
SkillRet-Embedding-0.6B	Original	55.68	66.04	73.26	53.20	58.11	60.81	26.42	44.03	52.83
	+ Ours	**63.55**	**74.70**	**78.58**	**60.16**	**65.33**	**66.72**	**36.48**	**56.60**	**59.75**
	Δ	+7.87	+8.66	+5.32	+6.96	+7.22	+5.91	+10.06	+12.57	+6.92
R3-Embedding	Original	54.70	66.79	71.04	53.42	59.30	60.97	25.16	46.54	50.94
	+ Ours	**66.39**	**75.59**	**77.53**	**62.47**	**66.82**	**67.54**	**39.62**	**56.60**	**59.75**
	Δ	+11.69	+8.80	+6.49	+9.05	+7.52	+6.57	+14.46	+10.06	+8.81

D.3 Generalization to SRA-Bench

To evaluate the generalizability of SkillDreamer across different skill retrieval benchmarks, we further conduct experiments on SRA-Bench. We apply SkillDreamer with Qwen3-Embedding-0.6B as the retrieval backbone and compare it with the original retriever. As shown in Table 13, SkillDreamer improves Recall@5 from 57.90 to 66.08, NDCG@5 from 52.93 to 61.58, and Completeness@5 from 51.43 to 59.19, yielding gains of 8.18, 8.65, and 7.76 points, respectively. The consistent improvements across these three metrics show that SkillDreamer enhances retrieval coverage, ranking quality, and the completeness of the retrieved skill set. Similar gains are observed at $K = 10$ and $K = 15$, further demonstrating its effectiveness and generalizability on SRA-Bench.

Table 13: Retrieval performance (%) on SRA-Bench with Qwen3-Embedding-0.6B. Best results are highlighted in **bold**.

Retriever	Setting	Recall			NDCG			Comp.		
		@5	@10	@15	@5	@10	@15	@5	@10	@15
Qwen3-Embedding-0.6B	Original	57.90	63.10	66.52	52.93	54.86	55.90	51.43	56.48	59.78
	+ Ours	**66.08**	**69.73**	**70.95**	**61.58**	**63.02**	**63.42**	**59.19**	**62.89**	**64.22**
	Δ	+8.18	+6.63	+4.43	+8.65	+8.16	+7.52	+7.76	+6.41	+4.44

D.4 Generalization across Diverse Backbones

To further verify the plug-and-play capability of SkillDreamer, we extend the evaluation to additional embedding backends, including BGE-large-en-v1.5 (Xiao et al., 2024), E5-large-v2 (Wang et al., 2022), and Qwen3-Embedding-8B (Zhang et al., 2025). As shown in 14–16, SkillDreamer consistently improves Recall, NDCG, and Completeness across all backbones and datasets, demonstrating its effectiveness in enhancing skill coverage and ranking quality across diverse retrieval backbones. These results further validate that SkillDreamer can be seamlessly integrated with different embedding models without relying on a specific retrieval backbone.

Table 14: Complete retrieval results (%) with additional embedding backbones on SkillUsage; per-backbone best results are in **bold**.

Retriever	Setting	Recall			NDCG			Comp.		
		@5	@10	@15	@5	@10	@15	@5	@10	@15
BGE-large-en-v1.5	Original	47.56	53.88	56.84	48.24	50.92	51.98	32.18	35.63	39.08
	+ Ours	**58.83**	**63.38**	**65.37**	**56.39**	**58.20**	**58.96**	**40.23**	**45.98**	**48.28**
	Δ	+11.27	+9.50	+8.53	+8.15	+7.28	+6.98	+8.05	+10.35	+9.20
E5-large-v2	Original	46.42	48.29	50.48	46.26	46.93	47.74	32.18	32.18	33.33
	+ Ours	**53.66**	**57.93**	**60.65**	**53.43**	**55.14**	**56.15**	**36.78**	**40.23**	**43.68**
	Δ	+7.24	+9.64	+10.17	+7.17	+8.21	+8.41	+4.60	+8.05	+10.35
Qwen3-Embedding-8B	Original	53.39	60.95	69.72	54.77	57.91	60.78	32.18	41.38	50.57
	+ Ours	**58.42**	**69.11**	**73.45**	**57.29**	**61.67**	**63.04**	**37.93**	**50.57**	**55.17**
	Δ	+5.03	+8.16	+3.73	+2.52	+3.76	+2.26	+5.75	+9.19	+4.60

Table 15: Complete retrieval results (%) with additional embedding backbones on SkillRet; per-backbone best results are in **bold**.

Retriever	Setting	Recall			NDCG			Comp.		
		@5	@10	@15	@5	@10	@15	@5	@10	@15
BGE-large-en-v1.5	Original	55.96	61.40	64.31	53.75	55.81	56.71	39.30	44.63	47.49
	+ Ours	**66.69**	**71.47**	**73.94**	**61.42**	**63.23**	**63.98**	**51.45**	**56.95**	**59.74**
	Δ	+10.73	+10.07	+9.63	+7.67	+7.42	+7.27	+12.15	+12.32	+12.25
E5-large-v2	Original	51.51	57.42	60.86	48.01	50.21	51.28	36.54	41.95	45.17
	+ Ours	**63.85**	**68.99**	**71.47**	**58.20**	**60.12**	**60.86**	**48.79**	**54.29**	**57.17**
	Δ	+12.34	+11.57	+10.61	+10.19	+9.91	+9.58	+12.25	+12.34	+12.00
Qwen3-Embedding-8B	Original	60.64	67.10	70.39	57.52	59.98	61.01	43.33	49.99	53.53
	+ Ours	**67.79**	**73.92**	**76.57**	**62.06**	**64.35**	**65.16**	**52.75**	**59.76**	**63.00**
	Δ	+7.15	+6.82	+6.18	+4.54	+4.37	+4.15	+9.42	+9.77	+9.47

Table 16: Complete retrieval results (%) with additional embedding backbones on SkillNet-Gym; per-backbone best results are in **bold**.

Retriever	Setting	Recall			NDCG			Comp.		
		@5	@10	@15	@5	@10	@15	@5	@10	@15
BGE-large-en-v1.5	Original	48.38	58.63	60.82	47.34	52.30	53.17	18.87	30.19	32.70
	+ Ours	**61.99**	**70.30**	**71.93**	**58.41**	**62.25**	**62.95**	**32.70**	**46.54**	**47.80**
	Δ	+13.61	+11.67	+11.11	+11.07	+9.95	+9.78	+13.83	+16.35	+15.10
E5-large-v2	Original	43.11	51.01	55.20	41.61	45.44	47.07	14.47	24.53	29.56
	+ Ours	**59.71**	**68.71**	**71.10**	**57.00**	**61.35**	**62.28**	**34.59**	**47.17**	**50.31**
	Δ	+16.60	+17.70	+15.90	+15.39	+15.91	+15.21	+20.12	+22.64	+20.75
Qwen3-Embedding-8B	Original	52.26	64.74	67.59	50.29	56.08	57.15	22.64	45.91	49.69
	+ Ours	**63.84**	**73.95**	**76.55**	**60.03**	**64.73**	**65.66**	**37.74**	**54.09**	**57.86**
	Δ	+11.58	+9.21	+8.96	+9.74	+8.65	+8.51	+15.10	+8.18	+8.17

D.5 Additional Ablation Results

Tables 17 and 18 extend the ablation study in the main paper by reporting Recall@K, NDCG@K, and Completeness@K at all evaluated cutoffs on SkillUsage and SkillRet. Consistent with the main results, the two pseudo-skill-only variants do not outperform Direct Retrieval, showing that prospective execution information alone cannot replace the original task semantics. In contrast, PSG + HSR performs best under every metric and cutoff, improving R@5 over Direct Retrieval by 7.74 and 7.79 points, NDCG@5 by 4.79 and 4.68 points, and Comp.@5 by 6.90 and 9.76 points on SkillUsage and SkillRet, respectively. These results further confirm the benefit of combining task semantics with capability-conditioned prospective information in hybrid skill retrieval.

Table 17: Ablation results (%) on SkillUsage. Best results are in **bold**.

Setting	Recall			NDCG			Comp.		
	@5	@10	@15	@5	@10	@15	@5	@10	@15
Direct Retrieval	52.68	62.25	67.66	52.62	56.51	58.36	35.63	43.68	48.28
Pseudo Skill w/o Cap.	48.51	58.69	63.75	49.29	53.74	55.37	31.03	37.93	43.68
Pseudo-Skill Retrieval	48.85	59.00	65.48	48.87	53.40	55.53	27.59	40.23	48.28
PSG + HSR	**60.42**	**69.47**	**70.57**	**57.41**	**60.83**	**61.33**	**42.53**	**51.72**	**52.87**

Table 18: Ablation results (%) on SkillRet. Best results are in **bold**.

Setting	Recall			NDCG			Comp.		
	@5	@10	@15	@5	@10	@15	@5	@10	@15
Direct Retrieval	59.14	64.91	68.20	56.21	58.40	59.42	41.83	47.21	50.47
Pseudo Skill w/o Cap.	49.56	55.15	57.91	46.54	48.59	49.44	33.20	37.62	40.00
Pseudo-Skill Retrieval	55.56	62.60	66.23	48.91	51.51	52.61	40.48	47.53	51.63
PSG + HSR	**66.93**	**71.73**	**74.48**	**60.89**	**62.69**	**63.52**	**51.59**	**56.95**	**60.16**

D.6 Additional Analysis on the Number of Skills

To further investigate the effect of the number of required skills, we extend the analysis to SkillRet. As shown in Fig. 4, SkillDreamer maintains comparable retrieval performance with raw query retrieval for single-skill queries. As the number of ground-truth skills increases, SkillDreamer increasingly outperforms raw query retrieval across R@5, R@10, and R@15, achieving larger performance gains on more complex multi-skill queries. These results further demonstrate the effectiveness of SkillDreamer in identifying diverse skills required for complex tasks.

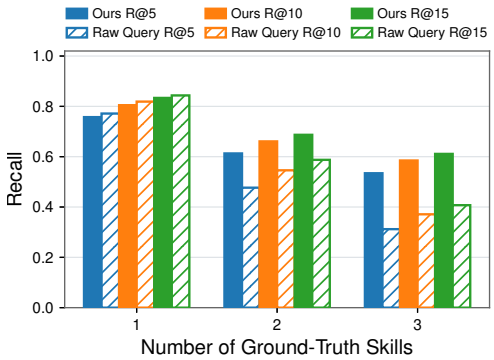


Figure 4: Recall@K on SkillRet across tasks with different numbers of ground-truth skills.

REFERENCES

Chishui Chen, Jiaye Lin, Te Sun, Yi Yang, Junxi Wang, Cong Qin, Yangen Hu, Lu Pan, and Ke Zeng. Skill or skip? learning selective skill invocation in agentic tasks via dual-granularity preference learning. *arXiv preprint arXiv:2606.00510*, 2026a.

Ruishuo Chen, Xun Wang, Yu Chen, Zhuoran Li, and Longbo Huang. The router within: Eliciting native skill routing from a frozen LLM. *arXiv preprint arXiv:2609.15982*, 2026b.

Yanping Chen, Weijie Shi, Wen Yang, and Jiajie Xu. Task decomposition-guided reranking for adaptive agent skill retrieval. *arXiv preprint arXiv:2607.06283*, 2026c.

Hongcheol Cho, Ryangkyung Kang, and Youngeun Kim. Skillret: A large-scale benchmark for skill retrieval in llm agents. *arXiv preprint arXiv:2605.05726*, 2026.

Jiandong Ding. Skillresolve-bench: Measuring and resolving same-capability ambiguity in agent skill retrieval. *arXiv preprint arXiv:2606.10388*, 2026.

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*, pp. 1762–1777, 2023.

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 2020 conference on empirical methods in natural language processing (EMNLP)*, pp. 6769–6781, 2020.

Fangzhou Li, Pagkratios Tagkopoulos, and Ilias Tagkopoulos. Skillflow: Scalable and efficient agent skill retrieval system. In *Deep Learning for Code: Towards Human-Centered Coding Agents*, 2026a.

Qingyao Li, Wenxiang Jiao, Shuai Shao, Kangning Zhang, Yuan Lu, Yi Guo, Weiwen Liu, Weinan Zhang, and Yong Yu. Skillgate: Training in-policy skill selection in long-horizon agents. *arXiv preprint arXiv:2608.18852*, 2026b.

Xiangyi Li, Yimin Liu, Wenbo Chen, Bingran You, Zonglin Di, Yifeng He, Shenghan Zheng, Kyoung Whan Choe, Jiankai Sun, Shuyi Wang, et al. Skillsbench: Benchmarking how well agent skills work across diverse tasks. *arXiv preprint arXiv:2602.12670*, 2026c.

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

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. *arXiv preprint arXiv:2604.04323*, 2026.

Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. Query rewriting in retrieval-augmented large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 5303–5315, 2023.

Zhen Qin, Rolf Jagerman, Kai Hui, Honglei Zhuang, Junru Wu, Le Yan, Jiaming Shen, Tianqi Liu, Jialu Liu, Donald Metzler, et al. Large language models are effective text rankers with pairwise ranking prompting. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 1504–1518, 2024.

Revanth Gangi Reddy, JaeHyeok Doo, Yifei Xu, Md Arafat Sultan, Deevya Swain, Avirup Sil, and Heng Ji. First: Faster improved listwise reranking with single token decoding. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 8642–8652, 2024.

Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Information Retrieval*, 3(4):333–389, 2009.

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

Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. Is chatgpt good at search? investigating large language models as re-ranking agents. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 14918–14937, 2023.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*, 2022.

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.

Zifei Wang, Wei Wen, Qiang Ji, Ruizhi Qiao, and Xing Sun. Skill is not document: A query-conditional benchmark and two-stage retriever for llm agent skill routing. *arXiv preprint arXiv:2606.03565*, 2026.

Wenhao Wu, Menghao Zhang, Xin Wang, Zhi Wang, Kun Shao, and Jian Luan. TRACE: A self-evolving skill bank for consistent, limit-aware LLM agents. *arXiv preprint arXiv:2608.22793*, 2026.

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.

Kun Zeng, Yu Huo, Siyu Zhang, Zi Ye, Yuecheng Zhuo, Haoyue Liu, Yuquan Lu, Junhao Wen, and Xiaoying Tang. Group of skills: Group-structured skill retrieval for agent skill libraries. *arXiv preprint arXiv:2605.06978*, 2026.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, et al. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025.

YanZhao Zheng, ZhenTao Zhang, Chao Ma, YuanQiang Yu, JiHuai Zhu, Yong Wu, Tianze Xu, Baohua Dong, Hangcheng Zhu, Ruohui Huang, et al. Skillrouter: Skill routing for llm agents at scale. *arXiv preprint arXiv:2603.22455*, 2026.