SKILLCOMP: INTERNALIZING SKILL RELATIONS IN DENSE RETRIEVERS FOR COMPLETE SKILL-SET RETRIEVAL FROM LARGE-SCALE LIBRARIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Reusable skills equip large language model agents with specialized capabilities for executing complex tasks, which often require multiple skills to work together. However, some skills needed in intermediate steps may be implicit in the task description, making it difficult for dense retrievers to retrieve the complete required skill set from a large skill library. To address this limitation, we propose **SkillComp**, which uses structural relations among required skills to construct additional training supervision for complete skill-set retrieval. Specifically, SkillComp selects related skills based on these relations and appends their descriptions to the query to form expanded queries, providing additional context for retrieving the remaining required skills. It then jointly trains the retriever on the plain and expanded queries using the complete required skill set as positive targets, thereby improving the alignment between the plain query and the complete required skill set. Experiments on SkillRet and SkillBench show that SkillComp improves skill retrieval, with particularly strong gains on multi-skill tasks. On the multi-skill subset of SkillRet, SkillComp improves Recall@10 and Completeness@10 over Qwen3-Embedding-0.6B by +33.08 and +48.54 points, respectively, and over the strongest baseline by +2.57 and +5.19 points. These results support that SkillComp improves complete-set retrieval, while retaining standard plain retrieval at inference.

1 INTRODUCTION

Reusable skills package procedures, tools, code, and domain knowledge into modular capabilities that LLM agents can invoke across tasks (Li et al., 2026; Wang et al., 2023). Given a task, an LLM agent relies on skill descriptions to bridge the gap between the high-level task goal and its execution. These descriptions specify what each skill does and what it consumes and produces, allowing the agent to route and orchestrate multiple skills into an execution workflow (Liu et al., 2026a; Wang et al., 2026b; Yao et al., 2026). As skill repositories grow to millions of entries[1], limited context windows make it impractical for LLM agents to load and reason over all available skills (Liu et al., 2024; Su et al., 2026). Skill retrieval, which filters and ranks candidates into a limited set, is therefore necessary for LLM agents (Liu et al., 2026b). Since the agent plans its workflow using the retrieved skills, missing any required skill may leave part of the workflow unsupported. The retrieval target is therefore the complete required skill set rather than a single best-matching skill (Lumer et al., 2025).

Recent skill retrievers adapt general-purpose dense encoders to the skill domain using contrastive objectives that optimize semantic similarity between task queries and skill descriptions (Kang et al., 2026; Zheng et al., 2026). For a multi-skill task, these methods form multiple positive query–skill pairs from the same task query, requiring the learned query representation to support multiple skills simultaneously (Khosla et al., 2020; Sidiropoulos & Kanoulas, 2024). However, some required skills support prerequisite or intermediate steps that are implied by other required skills rather than directly stated in the task description (Yao et al., 2026; Liu et al., 2026a). For example, an upstream skill may produce output in a specific format that requires a downstream skill to process it, even if this requirement is not explicitly stated in the task description.

[1] https://skillsmp.com/

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.

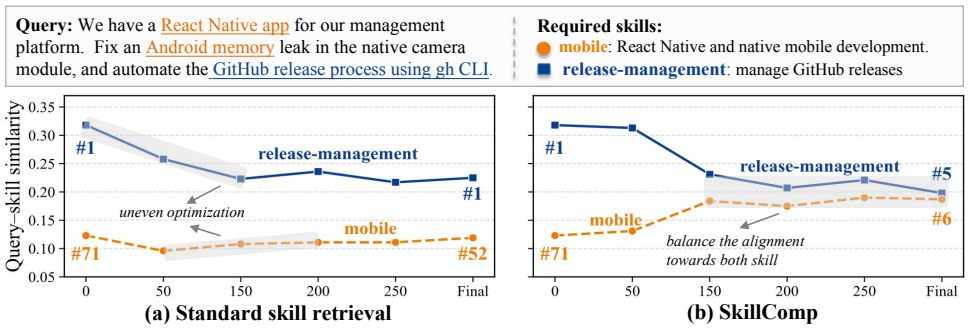


Figure 1: **Training trajectories.** The curves track the similarity between the plain query and each required skill during training, measured by the relative score R_τ (Appendix E.6); # denotes rank. Initially, `release-management` is highly ranked and closely aligned with the plain query, whereas `mobile` is much less aligned. (a) Standard training largely preserves this imbalance, with only limited improvement for `mobile`. (b) SkillComp preserves the strong alignment of `release-management` while substantially improving that of `mobile`, allowing both required skills to be highly ranked under the plain query.

Since required skills receive different levels of semantic match from the task query, their positive query–skill pairs may favor different update directions, causing some required skills to improve less than others (Sener & Koltun, 2018; Yu et al., 2020; Liu et al., 2021). As shown in Figure 1(a), a weakly expressed required skill can remain poorly aligned with the task query throughout standard training (Wang et al., 2026a; Chang, 2025). The retriever therefore recover only part of the required skill set, resulting in *incomplete retrieval* (Wang et al., 2026b).

To address this limitation, we propose **SkillComp**, which introduces additional training supervision from structural relations among required skills to improve the relevance of the complete required skill set to the plain task query. These relations can capture dependencies or co-required patterns among skills that may be weakly expressed in the task description (Lumer et al., 2025). SkillComp converts these structural relations into textual supervision by appending a description to the plain query, forming expanded queries (Xiong et al., 2020; Yu et al., 2021). For example, appending the upstream skill makes the requirement for its downstream skill explicit. In this way, the expanded queries provide additional cues while preserving the original task goal.

SkillComp then jointly trains the retriever on the plain and expanded queries. The expanded query is constructed from the plain query, so the two inputs remain closely related. Prior work suggests that such related inputs can produce compatible training signals (Clark et al., 2018; Xie et al., 2020). Our analysis further shows that, for a given required skill, positive gradient alignment between the expanded- and plain-query losses provides an additional reduction in the plain-query contrastive loss, thereby strengthening the alignment between the query and that skill. To make this direct benefit available to every required skill, SkillComp uses the complete required skill set as positive targets for both queries. At inference time, SkillComp uses only the plain query without query expansion.

We evaluate SkillComp on SkillRet (Kang et al., 2026) and SkillBench (Li et al., 2026) using two retrieval metrics: R@K (recall), and C@K (completeness). SkillComp achieves the highest R@20 and C@20 among the compared retrievers with both 0.6B and 8B backbones, while using only the plain query at inference. The gains are larger in completeness: on multi-skill queries, SkillComp improves C@10 over Qwen3-Embedding-0.6B (Zhang et al., 2025) by +48.54 and surpasses the strongest baseline, R3-Skill (Wang et al., 2026b), by +5.19. These results indicate that SkillComp mitigates incomplete retrieval by jointly optimizing the plain and expanded queries, allowing supervision from the expanded queries to improve retrieval from the plain query.

2 Preliminaries

We first formalize the skill retrieval problem and the objective used to train skill retrievers.

Skill retrieval. Given a skill collection $\mathcal{S}$ and a task described by a plain query q, let $G \subseteq \mathcal{S}$ denote the set of skills required to achieve the task goal (Su et al., 2026; Zheng et al., 2026). A dense retriever encodes the task and candidate skills as $z_q = E_q(q; \theta_q)$ and $z_s = E_s(s; \theta_s)$ using a query encoder $E_q(\cdot; \theta_q)$ and a skill encoder $E_s(\cdot; \theta_s)$, respectively (Karpukhin et al., 2020). Each candidate $s \in \mathcal{S}$

receives a relevance score $\mathrm{Sim}(z_q, z_s)$, typically computed as a dot product. Candidates are then ranked to form a list $L_K(q)$ containing the K highest-scoring skills. The retrieval goal is to include the complete required set G in $L_K(q)$, enabling the agent to access all necessary skills within its limited context (Liu et al., 2024; Kang et al., 2026; Qu et al., 2024).

Contrastive learning for skill retrieval. To achieve this retrieval goal, dense retrievers are typically trained with contrastive objectives, which maximize the similarity between task descriptions and their required skills relative to negative candidates (Xiong et al., 2021). Since a task may require one or more skills ($|G| \geq 1$), we adopt multi-positive supervision (Khosla et al., 2020; Wang et al., 2026a) and aggregate the per-skill losses $\ell(q, s; \mathcal{N}_q)$ over G as:

$$\mathcal{L}_{\mathrm{set}}(q; G) = \frac{1}{|G|} \sum_{s \in G} \ell(q, s; \mathcal{N}_q), \quad \ell(q, s; \mathcal{N}_q) = -\log \frac{e^{\mathrm{Sim}(z_q, z_s)/\tau}}{\sum_{k \in \mathcal{N}_q \cup \{s\}} e^{\mathrm{Sim}(z_q, z_k)/\tau}}, \tag{1}$$

where τ is the temperature and $\mathcal{N}_q$ is the negative set (van den Oord et al., 2018; Chen et al., 2020). All required skills are excluded from $\mathcal{N}_q$ to prevent false-negative penalties (Qu et al., 2021; Huynh et al., 2022). We denote the base plain-query objective as $\mathcal{L}_D(q; G) = \mathcal{L}_{\mathrm{set}}(q; G)$, where each required skill $s \in G$ is paired with the same plain query through an individual contrastive loss $\ell_s(\theta) := \ell(q, s; \mathcal{N}_q)$. Here, $\theta = (\theta_q, \theta_s)$ denotes the trainable parameters of both encoders (Appendix C).

3 Analysis of Multi-Skill Optimization

The plain-query objective supervises every required skill, but their losses can decrease at different rates. We analyze why this happens and when expanded-query supervision can improve their optimization under the plain query.

Gradient interactions among required skills. For each required skill $s \in G$, let $g_s = \nabla_\theta \ell_s(\theta)$ be the gradient of its plain-query loss, defined in Eq. 1. The training objective averages the losses of all $m = |G|$ required skills, so its gradient is $g_D = \nabla_\theta \mathcal{L}_D(q; G) = \frac{1}{m} \sum_{t \in G} g_t$. Each skill is therefore affected by the gradients of the other required skills. For a sufficiently small step size $\eta > 0$, the update $\theta - \eta g_D$ changes the loss of s by

$$\ell_s(\theta - \eta g_D) - \ell_s(\theta) = -\eta \langle g_s, g_D \rangle + o(\eta) = -\frac{\eta}{m} \left[\underbrace{\|g_s\|^2}_{\text{self}} + \underbrace{\textstyle\sum_{t \in G, t \neq s} \langle g_s, g_t \rangle}_{\text{cross-target}} \right] + o(\eta). \tag{2}$$

Here, $o(\eta)$ denotes terms that vanish faster than η as the step size approaches zero. The self term measures the contribution of s's own gradient to reducing its loss. The cross-target terms measure how the other skills' gradients help or oppose that reduction (Sener & Koltun, 2018; Yu et al., 2020; Liu et al., 2021). We denote their combined effect by $p_s = \langle g_s, g_D \rangle$, so the loss decreases by approximately ηp_s in one small step. A larger positive value means a greater loss reduction for the same small step size, while a negative value means that the shared update instead increases this skill's loss. Although all required skills receive equal weight in the objective, the shared update can help some skills more than others because their gradients differ (Wang et al., 2026a; Chang, 2025).

Expanded-query supervision. SkillComp adds supervision through expanded queries during training. We select required skills $A \subseteq G$ and append their descriptions to the task description, forming $q_A = q \oplus A$ (Xiong et al., 2020; Zhao et al., 2021). Section 4.1 describes how we select A. The plain and expanded queries provide two views of the same task, with the expanded view supplying additional skill context. Both views train the same retriever, so an update from the expanded-query loss can also change the loss under the plain query (Clark et al., 2018; Xie et al., 2020).

Transfer to plain-query optimization. Let H be the gradient of the auxiliary expanded-query loss. We compare two updates from the same parameters θ: the base update $\theta_B = \theta - \eta g_D$ and the joint update $\theta_A = \theta - \eta(g_D + H)$. Both use the same step size η, and both query views use the same negative set $\mathcal{N}_q$. All gradients are evaluated at θ. For a required skill s, first-order Taylor expansion gives the difference in its plain-query loss after the two updates:

$$\Delta_s(H) := \ell_s(\theta_B) - \ell_s(\theta_A) = \eta \langle g_s, H \rangle + o(\eta). \tag{3}$$

A positive $\Delta_s(H)$ means that adding expanded-query supervision gives s a lower plain-query loss than the base update. For sufficiently small η, this holds when $\langle g_s, H \rangle > 0$: the auxiliary gradient points in a direction that helps reduce the plain-query loss of s (Du et al., 2018).

The auxiliary gradient H combines contributions from the expanded-query targets. If s is one of these targets, its contribution comes from $h_s^{(A)} = \nabla_\theta \ell(q_A, s; \mathcal{N}_q)$. We call this a same-target contribution because g_s and $h_s^{(A)}$ are gradients for the same skill under the two query views. A positive $\langle g_s, h_s^{(A)} \rangle$ helps reduce the plain-query loss of s. Other targets can reinforce or offset this contribution, so the overall effect depends on $\langle g_s, H \rangle$.

Conditions for reducing uneven optimization. Adding H changes the progress for each skill from p_s to $p_s^{\text{aug}} = p_s + \langle g_s, H \rangle$. A positive auxiliary contribution improves on the base update. For the loss to decrease from its current value to first order, the combined quantity must be positive: $p_s^{\text{aug}} > 0$. Now consider two required skills with $p_s < p_t$, meaning that the base update makes less progress on s than on t. Their gap narrows if the auxiliary update contributes more to s, that is, $\langle g_s, H \rangle > \langle g_t, H \rangle$, provided their order remains unchanged after the update. Reducing uneven optimization therefore depends on the relative contributions to different skills, as well as the improvement for any one skill. The expanded queries and their positive targets determine these contributions through H. Section 4 describes these choices, and Appendix E provides the derivations.

4 SKILLCOMP

Building on the analysis above, we propose **SkillComp**. SkillComp uses relations among required skills to construct expanded queries and jointly trains on the plain and expanded queries using the complete required skill set as positive targets.

4.1 RELATION-GUIDED EXPANDED QUERY CONSTRUCTION

Section 3 gives conditions under which expanded-query supervision benefits a required skill. To choose which skill description to append, we use relations among required skills. Prior work shows that relations among tools and skills, such as execution dependencies, can reveal task requirements that are not explicit in the plain query (Shen et al., 2023; Liu et al., 2026a; Qu et al., 2024; Yao et al., 2026; Lumer et al., 2025). We therefore use an LLM to annotate pairwise relations among the required skills before retriever training, then select the appended skill r based on these relations.

We consider two types of relations. For a *sequential dependency* ($s_i \rightarrow s_j$), s_i is a prerequisite or provides an input required by s_j. We choose the upstream skill s_i as r. Its description adds prerequisite or input information that may be absent from the plain query, making the need for the downstream skill s_j more explicit (Liu et al., 2026a; Yao et al., 2026). For a *co-required relation* ($s_i \parallel s_j$), the task requires both skills without a fixed execution order. Appending either skill's description allows the expanded view to represent this joint requirement. Because this relation does not determine which skill to append, we choose as r the skill with lower similarity to the plain query under the current retriever. Appendix A.2 shows that SkillComp is not highly sensitive to this choice.

In both cases, we append the selected skill descriptions cumulatively to form expanded queries $q_A = q \oplus A$ (Appendix F). For example, for a three-skill task with a sequential workflow, a second expanded query $q \oplus r \oplus r'$ further appends a skill r' that is adjacent to r in the dependency relations. This follows the same rationale as above: $q \oplus r$ provides context for the skill that directly depends on r, while appending r' extends this context to the next step in the workflow.

4.2 JOINT TRAINING OBJECTIVE

Complete-set supervision. We next choose which required skills supervise each expanded query q_A. Consider any nonempty target set $T \subseteq G$. For each $t \in T$, let $h_t^{(A)} = \nabla_\theta \ell(q_A, t; \mathcal{N}_q)$ and $h_T^{(A)} = \frac{1}{|T|} \sum_{t \in T} h_t^{(A)}$. Its inner product with the plain-query gradient of a required skill s is

$$\langle g_s, h_T^{(A)} \rangle = \frac{1}{|T|} \Big[\underbrace{\mathbf{1}[s \in T] \, \langle g_s, h_s^{(A)} \rangle}_{\text{same-target}} + \underbrace{\sum_{t \in T \setminus \{s\}} \langle g_s, h_t^{(A)} \rangle}_{\text{cross-target}} \Big]. \tag{4}$$

The same-target term, defined in Section 3, is present only when $s \in T$; otherwise, the auxiliary effect on s depends entirely on cross-target interactions. We therefore supervise each expanded query with the complete required skill set, $T = G$, which keeps the appended skills in A as positive targets and preserves the same-target term for every required skill. Appendix E provides further analysis.

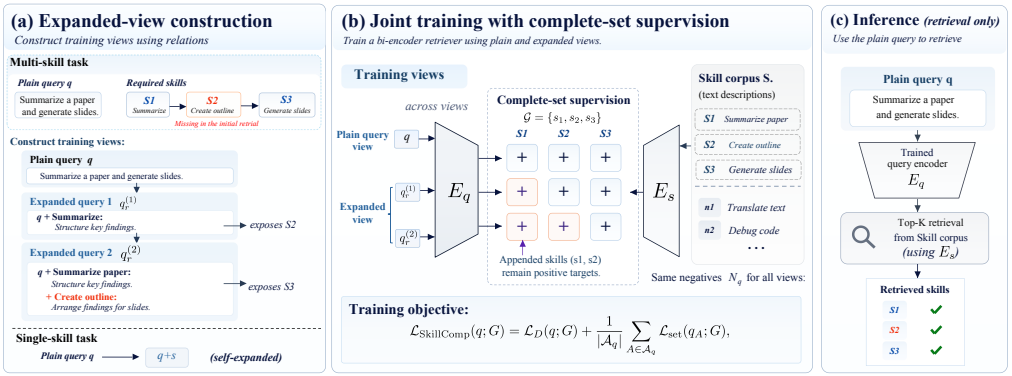


Figure 2: **Overview of SkillComp.** (a) Skill relations guide the selection of required skills, whose descriptions are appended cumulatively to the plain query q to form the expanded queries. (b) The plain and expanded queries train the same retriever against the complete required skill set G. Inference uses only q.

SkillComp objective. For multi-skill tasks, we construct expanded queries using the relations described in Section 4.1. Single-skill tasks have no skill relations and would otherwise receive no expanded-query supervision. For such a task, we append its sole required skill to form $q_s = q \oplus s$ and supervise s under both views. The auxiliary gradient then contains only a same-target term, which lowers the plain-query loss relative to the base update when this term is positive and the step size is sufficiently small (Section 3).

Let $\mathcal{A}_q$ denote the appended skill sets for q; each $A \in \mathcal{A}_q$ defines an expanded query q_A, and every task has at least one. SkillComp adds the average expanded-query loss to the plain-query loss:

$$\mathcal{L}_{\text{SkillComp}}(q; G) = \mathcal{L}_D(q; G) + \frac{1}{|\mathcal{A}_q|} \sum_{A \in \mathcal{A}_q} \mathcal{L}_{\text{set}}(q_A; G). \tag{5}$$

Averaging over $\mathcal{A}_q$ gives each task the same weight in the auxiliary objective regardless of how many expanded queries it has, and the plain-query and averaged expanded-query losses have equal weight. Both views use the same negative set $\mathcal{N}_q$, which excludes all required skills in G. Expanded queries and skill relations are used only during training, as privileged information (Vapnik & Vashist, 2009); at inference, SkillComp performs standard dense retrieval with the plain query alone.

Local gradient compatibility. We examine how the auxiliary gradient affects the plain-query objective on 768 two-skill tasks from two batches. Let g_t and h_t denote the gradients for target t under q and q_r, respectively. The plain-query and auxiliary gradients are then $g_D = (g_r + g_s)/2$ and $h_G = (h_r + h_s)/2$. We compute all gradients at training step 50 with respect to the query encoder. Each task has one expanded query q_r, where r is the appended skill and s is the other required skill. Figure 3a shows positive same-target alignment for every evaluated task, while cross-target inner products are negative in most tasks. By Eq. 4, positive same-target alignment adds to the first-order decrease in each skill's plain-query loss, supporting our choice to retain every required skill as a positive target. Figure 3b shows that the joint update gives a larger first-order decrease in the mean plain-query loss than the base update for 765 of 768 tasks (99.6%) at the same step size. Appendix B reports each skill's alignment with the base gradient and compares complete-set supervision with supervision that excludes the appended skill.

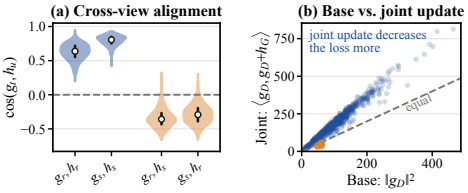


Figure 3: Local gradient compatibility. (a) Same-target and cross-target cross-view alignment. (b) First-order decrease in the aggregate plain-query loss under the base and joint updates.

5 Experiment

5.1 Experimental Setup

Baselines. We compare SkillComp with general-purpose and task-specific retrievers. General-purpose baselines include BM25 (Robertson & Zaragoza, 2009), BGE-small-en-v1.5, E5-large-v2,

Table 1: Retrieval results on SkillRet and SkillBench benchmarks. Best results are in bold.

Qwen3-Embedding-0.6B									
Benchmark	Model	Params	Hit@1	R@10	R@15	R@20	C@10	C@15	C@20
SkillRet	BM25	–	47.87	56.55	59.95	62.26	41.09	44.27	46.63
	bge-small-en-v1.5	33M	52.99	57.50	60.40	62.34	40.74	43.57	45.37
	e5-large-v2	335M	48.51	57.39	60.86	63.28	41.93	45.15	47.53
	Qwen3-Embedding-0.6B	0.6B	58.82	64.99	68.17	70.70	47.43	50.49	53.15
	SkillRet-Embedding-0.6B	0.6B	61.80	70.59	73.40	75.41	55.93	59.36	61.82
	SkillRouter-Embedding-0.6B	0.6B	73.12	75.69	78.26	80.23	59.09	62.14	64.60
	R3-Skill	0.6B	**80.31**	86.94	89.51	91.15	76.37	80.37	83.01
	SkillReason-Embedding-0.6B	0.6B	72.38	75.62	78.46	80.38	58.74	62.52	64.92
	SkillComp-Embedding-0.6B	0.6B	79.25	**87.38**	**90.08**	**91.90**	**78.29**	**82.43**	**85.23**
SkillBench	BM25	–	58.67	57.87	63.03	67.47	42.67	48.00	53.33
	bge-small-en-v1.5	33M	56.00	54.91	56.58	58.23	37.33	38.67	40.00
	e5-large-v2	335M	54.67	55.55	56.35	56.95	40.00	40.00	40.00
	Qwen3-Embedding-0.6B	0.6B	58.67	59.98	61.95	64.15	44.00	45.33	46.67
	SkillRet-Embedding-0.6B	0.6B	57.33	62.75	66.75	67.61	44.00	49.33	50.67
	SkillRouter-Embedding-0.6B	0.6B	66.67	69.12	72.80	76.22	49.33	54.67	56.00
	R3-Skill	0.6B	68.00	**71.00**	74.09	75.95	52.00	57.33	61.33
	SkillReason-Embedding-0.6B	0.6B	69.33	68.08	72.68	74.50	**53.33**	58.67	61.33
	SkillComp-Embedding-0.6B	0.6B	**72.00**	70.50	**78.27**	**80.29**	**53.33**	**61.33**	**66.67**
Qwen3-Embedding-8B									
Benchmark	Model	Params	Hit@1	R@10	R@15	R@20	C@10	C@15	C@20
SkillRet	Qwen3-Embedding-8B	8B	59.38	65.93	69.23	71.49	49.09	52.77	55.49
	SkillRet-Embedding-8B	8B	80.23	90.59	92.87	94.14	83.71	87.21	89.39
	SkillComp-Embedding-8B	8B	**84.23**	**92.53**	**94.22**	**95.54**	**86.45**	**89.49**	**91.57**
SkillBench	Qwen3-Embedding-8B	8B	64.00	67.78	71.56	74.67	50.67	54.67	57.33
	SkillRet-Embedding-8B	8B	60.00	70.90	75.91	78.89	**54.67**	61.33	**64.00**
	SkillComp-Embedding-8B	8B	**66.67**	**72.32**	**80.00**	**80.26**	**54.67**	**64.00**	**64.00**

Table 2: Retrieval results on the multi-skill subsets of SkillRet and SkillBench. Best results are in bold.

	SkillRet				SkillBench			
Model	R@10	R@20	C@10	C@20	R@10	R@20	C@10	C@20
Qwen3-Embedding-0.6B	50.05	57.24	17.48	24.50	54.87	61.00	31.37	37.25
SkillRet-Embedding-0.6B	63.79	69.99	36.45	44.62	56.98	62.17	29.41	37.25
SkillRouter-Embedding-0.6B	63.71	69.73	32.75	40.63	62.43	68.95	33.33	39.22
R3-Skill	80.56	86.44	60.83	71.25	65.20	72.47	37.25	50.98
SkillReason-Embedding-0.6B	62.24	68.69	30.73	39.84	64.82	70.34	**43.14**	50.98
SkillComp-Embedding-0.6B	**83.13**	**88.95**	**66.02**	**76.51**	**66.41**	**74.93**	41.18	**54.90**

and Qwen3-Embedding (Zhang et al., 2025). Task-specific baselines include SkillRet (Kang et al., 2026), SkillRouter (Zheng et al., 2026), R3-Skill (Wang et al., 2026b), and SkillReason (Jiang et al., 2026a), each fine-tuned on the training data specified in its paper. For R3-Skill, we evaluate its embedding checkpoint. SkillComp uses the same SkillRet training data as SkillRet. We evaluate the official checkpoints of all baselines under the same protocol.

Benchmark. We evaluate on two skill retrieval benchmarks. SkillRet (Kang et al., 2026) contains 4,997 test queries, 2,678 of which require multiple skills, over 6,660 candidate skills. SkillBench (Li et al., 2026; Zheng et al., 2026) contains 75 queries, 51 of which require multiple skills, over a candidate corpus of about 80K skills.

Evaluation metrics. Throughout the paper, we abbreviate Recall as R and Completeness as C and report $R@K$ and $C@K$ for $K \in \{10, 15, 20\}$. $R@K$ measures the fraction of required skills retrieved, averaged over queries, while $C@K$ measures the fraction of queries for which the complete required set is retrieved within the Top-K candidates (Qu et al., 2024). Hit@1 is the fraction of queries whose top-ranked skill is required. All reported gains are in percentage points.

Training details. We use Qwen3-Embedding (0.6B and 8B) (Zhang et al., 2025) as backbone models with in-batch negatives (Karpukhin et al., 2020) and curated hard negatives. Training follows two stages. We first fine-tune the general-purpose embedding encoder with the standard objective $\mathcal{L}_D$ to adapt it to the skill retrieval domain. We then freeze the backbone and optimize separate LoRA adapters (Hu et al., 2022) for the query and skill encoders using the joint SkillComp objective, so that each encoder can adapt to its own inputs. Appendix A.4 provides further implementation details.

Table 3: Ablation of SkillComp on SkillRet with Qwen3-Embedding-0.6B as the backbone. *Base* follows the same two-stage training but uses only the plain-query loss $\mathcal{L}_D$, without expanded queries. *Rel.-guided*: relation-guided selection of the appended skill (vs. random), from §4.1. *Complete-set*: complete-set supervision for each expanded query q_A, from Eq. 4. *Cross-view*: expanded-query supervision applied to all tasks, including the self-expanded query for single-skill tasks, from §4.2.

Variant	Rel.-guided	Complete-set	Cross-view	R@10	R@20	C@10	C@20
Qwen3-Embedding-0.6B	–	–	–	64.99	70.70	47.43	53.15
Base	–	–	–	85.76	90.87	75.45	83.49
w/o Appended Supervision	✓	×	×	81.88	86.74	69.04	76.49
w/o Single-Skill Expansion	✓	✓	×	85.95	90.80	75.65	83.47
Random Appended Skill	×	✓	✓	87.00	91.33	77.77	84.65
SkillComp	✓	✓	✓	**87.38**	**91.90**	**78.29**	**85.23**

5.2 Skill Retriever Performance

SkillComp improves skill retrieval. Table 1 shows that SkillComp improves R@20 and C@20 over its backbone on both benchmarks, with gains persisting as the backbone scales from 0.6B to 8B. On SkillRet, SkillComp-Embedding-0.6B improves over its backbone by +21.20 and +32.08, and over the strongest 0.6B baseline, R3-Skill, by +0.75 and +2.22, respectively. At the 8B scale, SkillComp-Embedding-8B improves over its backbone by +24.05 and +36.08. The same trend holds on SkillBench, whose candidate skill corpus is substantially larger than that of SkillRet. SkillComp achieves the highest R@20 across both backbone sizes, outperforming R3-Skill by +4.34 at the 0.6B scale and SkillRet-Embedding-8B by +1.37 at the 8B scale. Its C@20 on SkillBench improves over R3-Skill by +5.34 at the 0.6B scale and matches SkillRet-Embedding-8B at the 8B scale. These improvements across backbone sizes and candidate corpus scales suggest that the gains are not limited to a particular model capacity or retrieval setting.

SkillComp yields larger completeness gains on multi-skill queries. We further evaluate queries that require multiple skills, where retrieving the complete set is important for successful execution. SkillComp achieves the highest R@10, R@20, and C@20 on both benchmarks in Table 2. Its improvements over the backbone are larger on these queries than over all queries in Table 1. On SkillRet, for example, R@20 and C@20 improve by +31.71 and +52.01 on multi-skill queries, compared with +21.20 and +32.08 across all queries. Compared with R3-Skill, the gain in the fraction of queries with all required skills retrieved is larger than the gain in average recall. On SkillBench, the corresponding improvements are +3.93 in C@10 and +1.21 in R@10, although SkillReason achieves a higher C@10 than SkillComp. These comparisons with R3-Skill suggest that the gains are concentrated on completing partially retrieved skill sets rather than uniformly increasing the number of retrieved required skills.

5.3 Ablation Study

Complete-set supervision improves joint optimization across query views. Unlike multi-hop retrieval (Xiong et al., 2020; Zhao et al., 2021), which excludes appended context from subsequent supervision, SkillComp retains the appended skills as positive targets. We therefore test whether preserving the corresponding same-target terms in Eq. 4 benefits joint training. We use the *w/o Appended Supervision* variant for this ablation. It uses $\{G \setminus A\}$ as the target set for each expanded query q_A, so the appended skills appear in q_A but receive no direct supervision under the expanded-query loss. Compared with Base, R@20 and C@20 decrease by 4.13 and 7.00, respectively, showing that expanded-query supervision alone does not necessarily improve retrieval. According to Eq. 4, excluding the appended skills removes their same-target terms, leaving their auxiliary effects determined by cross-target interactions, which may weaken their optimization.

Single-skill expansion further improves retrieval through same-target supervision. We use the *w/o Single-Skill Expansion* variant to test whether cross-view supervision should also be applied to single-skill tasks. This variant applies expanded-query supervision only to multi-skill tasks, whereas SkillComp additionally constructs a self-expanded query $q_s = q \oplus s$ for each single-skill task (§4.2). Extending cross-view supervision to single-skill tasks improves R@20 by +1.10 and C@20 by +1.76. Because q_s shares the same target s with the plain query, the cross-target sum in Eq. 4 vanishes, leaving only the same-target term. The improvement suggests that this additional same-target supervision also benefits training on single-skill tasks.

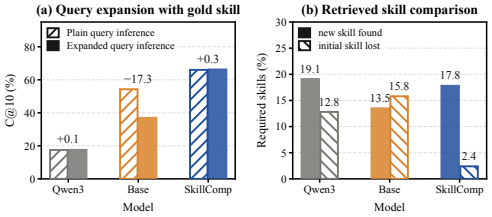


Figure 4: Query expansion with a gold skill on the SkillRet multi-skill subset. Each model first retrieves with the plain query q and then with $q \oplus r$, where r is the highest-ranked required skill in SkillComp's plain-query ranking and is shared across models. (a) C@10 before and after expansion. (b) Found is the percentage of required skills missed by the plain query that are retrieved after expansion; Lost is the percentage of required skills retrieved by the plain query that are no longer retrieved. (Appendix D).

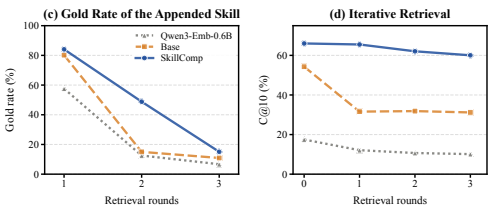


Figure 5: Iterative retrieval on the SkillRet multi-skill subset. We simulate test-time query expansion using each model's own Top-1 retrieved skill. Starting from the plain query at round 0, each subsequent round appends the Top-1 skill retrieved from the current query. Appended descriptions accumulate across rounds, and previously appended skills remain retrievable because retrieval targets the complete required set. (c) Percentage of queries whose appended skill is required. (d) C@10 at each round under the current expanded query.

Relation-guided expansion improves over random skill selection. We use the *Random Appended Skill* variant to test whether relation-guided selection is more effective than random selection of the appended skill. This variant samples r uniformly from G and periodically resamples it during training, whereas SkillComp follows the relation-based construction rule. The two variants use the same objective weights, with the random variant sampling a single r for each task. Relation-guided selection improves R@20 by +0.57 and C@20 by +0.58 over random selection. This comparison evaluates the combined effect of the relation-based construction rules described in §4.1.

5.4 Analysis

SkillComp combines information from both query views. SkillComp supervises the plain query q and its expanded queries q_A with the same required set G (§4.2). The plain query provides the original task information, while each expanded query adds the descriptions of selected required skills. If these two sources are integrated effectively, appending a required skill should recover missing skills without disrupting those already retrieved from the plain query. Figure 4 tests this behavior through inference-time query expansion on the SkillRet multi-skill subset Carpineto & Romano (2012). With a gold skill appended, SkillComp Lost and Found rates are 2.4% and 17.8%, respectively, excluding the appended skill (Appendix D). In contrast, Base loses 15.8% of its initially retrieved required skills and decreases by 17.32 in C@10. This suggests that SkillComp incorporates the additional expanded-query context while preserving information learned from the plain query.

Iterative test-time expansion does not improve over the plain query. We then replace the gold anchor with each model's own Top-1 retrieved skill, which more closely reflects test-time query expansion based on retrieved results, and repeat the expansion over three rounds. After the first round, SkillComp's C@10 changes little (66.02 to 65.50), whereas Base drops from 54.29 to 31.59 although most of its anchors are also correct (Hit@1 of 80.2% vs. 84.1%). When the expansion is repeated, however, the percentage of appended skills that are required falls to 15.1% by the third round and C@10 decreases to 60.04, consistent with query drift accumulating across rounds (Carpineto & Romano, 2012; Li et al., 2023). Despite this degradation, SkillComp remains more stable than the compared retrievers, supporting the use of expanded queries during training while retaining plain-query retrieval at inference.

SkillComp improves coverage over inference-time graph propagation. Skill relations can be incorporated either during training or directly at inference. Graph-based methods use a skill graph to propagate relevance and refine retrieval results at inference time. SkillComp instead uses skill relations to construct expanded queries during training while retaining standard dense retrieval at inference. We compare SkillComp with GoS (Liu et al., 2026a), a representative graph-based method. GoS requires a typed skill graph over the candidate library, so we evaluate it on SkillBench-2000, the 2K-skill library for which this graph is available.

Table 4: Comparison with graph-based retrieval on SkillBench-2000 (2K-skill library (Li et al., 2026), distinct from the 80K-skill main setting). MRR@10 assigns zero when no required skill is retrieved.

Method	R@5	R@10	C@5	C@10	MRR@10
Qwen3-Embedding-0.6B	62.42	69.61	43.68	54.02	66.44
+ GoS	62.00	63.97	43.68	45.98	71.46
text-embedding-3-large	64.30	67.70	47.13	52.87	72.36
+ GoS	64.18	68.04	44.83	55.17	**73.75**
SkillComp	**64.95**	**74.89**	**48.28**	**62.07**	70.93

We use Qwen3-Embedding-0.6B as the base retriever for GoS. GoS propagates Qwen retrieval scores over the skill graph at inference, whereas SkillComp is initialized from the same Qwen backbone, fine-tuned on SkillRet, and incorporates skill relations during training. We also evaluate text-embedding-3-large with and without GoS, following the embedding model used by GoS, to assess graph propagation with a different base retriever. As shown in Table 4, GoS improves first-hit ranking, increasing MRR@10 by 5.02 and 1.39 with the two base retrievers. Its effect on complete-set retrieval is less consistent: with Qwen3-Embedding-0.6B, GoS reduces R@10 by 5.64 and C@10 by 8.04. By comparison, SkillComp achieves the highest R@10 and C@10. In this comparison, inference-time graph propagation improves early ranking without consistently improving coverage of the complete required set, whereas SkillComp improves required-skill coverage using standard retrieval from the plain query. Furthermore, SkillComp uses skill relations only during training, avoiding graph maintenance as the skill corpus grows or evolves.

6 Related Work

Skill retrieval. Reusable skills package procedures, tools, code, and domain knowledge into modular capabilities that LLM agents can invoke across tasks (Jiang et al., 2026b; Si et al., 2026). Using external skills involves retrieval, incorporation, and application (Su et al., 2026); we focus on retrieval, which ranks a large skill repository into a short candidate list for the agent. SkillRouter and SkillRet fine-tune dense encoders and substantially outperform general-purpose retrievers (Zheng et al., 2026; Kang et al., 2026). Because a task often requires multiple skills, later work further models relations among them. In tool retrieval, COLT and Tool Graph Retriever encode tool co-occurrence or dependency into tool representations through graph learning (Qu et al., 2024; Gao et al., 2025). In skill retrieval, R3-Skill and SkillReranker rerank candidates from a dense retriever using query-conditioned compatibility or task decomposition (Wang et al., 2026b; Chen et al., 2026), while graph-based methods propagate relevance from initially retrieved skills over a prebuilt skill graph to expand or rerank the candidate set (Lumer et al., 2025; Liu et al., 2026a; Yao et al., 2026). These skill-retrieval methods depend on the output of a first-stage retriever, so their ability to recover missing required skills is partly constrained by the initial candidate set and the available graph connections. SkillComp instead uses skill relations to train the first-stage dense retriever itself.

Multi-positive retrieval. Many queries require multiple supporting items whose information is distributed across documents (Feldman & El-Yaniv, 2019; Qi et al., 2019; Asai et al., 2020). Multi-hop retrievers collect such items sequentially by conditioning each step on previously retrieved evidence (Xiong et al., 2020; Zhao et al., 2021; Fang & Glass, 2026). For single-step dense retrieval, recent work trains with multiple positives per query and explores how to aggregate, weight, or compare them within the contrastive objective to better balance supervision across relevant items (Sidiropoulos & Kanoulas, 2024; Chang, 2025; Wang et al., 2026a; Kim & Baek, 2025). These methods mainly control how much each positive contributes to the query-level objective, while still treating the positives as separate relevance targets. In skill retrieval, however, required skills may be connected through sequential dependencies, so one required skill can provide context for identifying another (Liu et al., 2026a; Yao et al., 2026). SkillComp uses these relations to construct expanded training queries and jointly supervises the plain and expanded queries with the complete required skill set.

7 Conclusion

We propose SkillComp, which introduces auxiliary supervision to improve complete required-skill retrieval. SkillComp uses relations among required skills to construct expanded queries and jointly supervises the plain and expanded queries with the complete required skill set. Experiments on SkillRet and SkillBench show improvements in R@20 and C@20 over the compared retrievers, with ties in some settings and particularly strong completeness gains on SkillRet multi-skill queries. These results demonstrate that incorporating structural information during training can improve coverage of the required skill set.

AI USE STATEMENT

In accordance with the ICLR Generative AI Policy, we disclose the use of generative AI tools in this work. Large language models (LLMs) were used in the following capacities:

- **Writing and polishing.** LLMs were used to edit and refine the manuscript's wording, improve syntactic fluency, and assist with organization.
- **Research ideation and theoretical assistance.** LLMs were consulted during research ideation and mathematical exploration related to the gradient alignment analysis in Section 3 and Appendix E.
- **Synthetic data construction.** An LLM was used in the data construction pipeline to annotate pairwise inter-skill relations for constructing expanded queries. As detailed in Appendix F, these annotations underwent multi-stage quality control, including automated consistency checks, secondary LLM audits, and manual inspection by the authors.

All AI-assisted content and artifacts were reviewed and verified by the authors. The authors take full responsibility for the final work, including its theoretical claims, methodology, experimental results, and written content.

REFERENCES

Akari Asai, Kazuma Hashimoto, Hannaneh Hajishirzi, Richard Socher, and Caiming Xiong. Learning to retrieve reasoning paths over wikipedia graph for question answering. In *International Conference on Learning Representations*, 2020. URL https://arxiv.org/abs/1911.10470.

Claudio Carpineto and Giovanni Romano. A survey of automatic query expansion in information retrieval. *ACM Computing Surveys*, 44(1):1–50, 2012. doi: 10.1145/2071389.2071390. URL https://doi.org/10.1145/2071389.2071390.

Shuai Chang. Improving dense passage retrieval with multiple positive passages, 2025. URL http://arxiv.org/abs/2508.09534.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 1597–1607. PMLR, 2020. URL https://proceedings.mlr.press/v119/chen20j.html.

Yanping Chen, Weijie Shi, Wen Yang, and Jiajie Xu. Task decomposition-guided reranking for adaptive agent skill retrieval. *arXiv preprint arXiv:2607.06283*, 2026. URL https://arxiv.org/abs/2607.06283.

Kevin Clark, Minh-Thang Luong, Christopher D. Manning, and Quoc Le. Semi-supervised sequence modeling with cross-view training. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 1914–1925, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1217. URL https://aclanthology.org/D18-1217/.

Yunshu Du, Wojciech M. Czarnecki, Siddhant M. Jayakumar, Mehrdad Farajtabar, Razvan Pascanu, and Balaji Lakshminarayanan. Adapting auxiliary losses using gradient similarity. *arXiv preprint arXiv:1812.02224*, 2018. doi: 10.48550/arXiv.1812.02224. URL https://arxiv.org/abs/1812.02224.

Wei Fang and James Glass. Beyond single-shot: Multi-step tool retrieval via query planning. *arXiv preprint arXiv:2601.07782*, 2026. URL https://arxiv.org/abs/2601.07782.

Yair Feldman and Ran El-Yaniv. Multi-hop paragraph retrieval for open-domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 2296–2309. Association for Computational Linguistics, 2019. doi: 10.18653/v1/P19-1222. URL https://aclanthology.org/P19-1222/.

Linfeng Gao, Yaoxiang Wang, Minlong Peng, Jialong Tang, Yuzhe Shang, Mingming Sun, and Jinsong Su. Tool graph retriever: Exploring dependency graph-based tool retrieval for large language models, 2025. URL http://arxiv.org/abs/2508.05152.

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.

Tri Huynh, Simon Kornblith, Matthew R. Walter, Michael Maire, and Maryam Khademi. Boosting contrastive self-supervised learning with false negative cancellation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 2785–2795, 2022. doi: 10.1109/wacv51458.2022.00106. URL https://openaccess.thecvf.com/content/WACV2022/html/Huynh_Boosting_Contrastive_Self-Supervised_Learning_With_False_Negative_Cancellation_WACV_2022_paper.html.

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*, 2026a. doi: 10.48550/arXiv.2608.08640. URL https://arxiv.org/abs/2608.08640.

Yanna Jiang, Delong Li, Hai Deng, Baihe Ma, Xu Wang, Qin Wang, and Guangsheng Yu. Sok: Agentic skills - beyond tool use in llm agents. *ArXiv*, abs/2602.20867, 2026b. URL https://api.semanticscholar.org/CorpusID:286001509.

Ryangkyung Kang, Hongcheol Cho, and Youngeun Kim. Skillret: A large-scale benchmark for skill retrieval in llm agents. *arXiv preprint arXiv:2605.05726*, 2026. doi: 10.48550/arXiv.2605.05726. URL https://arxiv.org/abs/2605.05726.

Vladimir Karpukhin, Barlas Oğuz, 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*, pp. 6769–6781. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-main.550. URL https://aclanthology.org/2020.emnlp-main.550/.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

Minsang Kim and Seung Jun Baek. Syntriever: How to train your retriever with synthetic data from LLMs. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 2523–2539, Albuquerque, New Mexico, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-naacl.136. URL https://aclanthology.org/2025.findings-naacl.136/.

Hang Li, Ahmed Mourad, Shengyao Zhuang, Bevan Koopman, and Guido Zuccon. Pseudo relevance feedback with deep language models and dense retrievers: Successes and pitfalls. *ACM Transactions on Information Systems*, 41(3):1–40, 2023. doi: 10.1145/3570724. URL https://doi.org/10.1145/3570724.

Xiangyi Li, Wen-Bo Chen, Yimin Liu, Sheng-Han Zheng, Xiaokun Chen, Yifeng He, Yubo Li, B. You, Haotian Shen, Jiankai Sun, Shuyi Wang, Qun-Hong Zeng, Di Wang, Xuandong Zhao, Xiaojun Wu, Yuanli Wang, Roey Ben Chaim, Zong-Lin Di, Yi Gao, Jun-Wei He, Yizhuo He, Liqiang Jing, Luyang Kong, Xin Lan, Jia-Chen Li, Songlin Li, Yijiang Li, Yue Lin, Xinyi Liu, Xuan-Qing Liu, Hao Lyu, Ze-Xiong Ma, Bowei Wang, Runhui Wang, Tianyu Wang, Wengao Ye, Yue Zhang, Boqin Yuan, Hanwen Xing, Yingying Xue, Steven Dillmann, and Han Lee. Skillsbench: Benchmarking how well agent skills work across diverse tasks. *arXiv preprint arXiv:2602.12670*, 2026. doi: 10.48550/arXiv.2602.12670. URL https://arxiv.org/abs/2602.12670.

Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. Conflict-averse gradient descent for multi-task learning. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

Dawei Liu, Zongxia Li, Hongyang Du, Xiyang Wu, Shihang Gui, Yongbei Kuang, and Lichao Sun. Graph-of-skills: Dependency-aware structural retrieval for massive agent skills. *arXiv preprint*

arXiv:2604.05333, 2026a. doi: 10.48550/arXiv.2604.05333. URL https://arxiv.org/abs/2604.05333.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024. doi: 10.1162/tacl_a_00638. URL https://aclanthology.org/2024.tacl-1.9/.

Yujian Liu, Jiabao Ji, Li An, T. Jaakkola, Yang Zhang, and Shiyu Chang. How well do agentic skills work in the wild: Benchmarking llm skill usage in realistic settings. *ArXiv*, abs/2604.04323, 2026b. URL https://api.semanticscholar.org/CorpusID:287201318.

Elias Lumer, Pradeep Honaganahalli Basavaraju, Myles Mason, James A. Burke, and Vamse Kumar Subbiah. Graph RAG-tool fusion, 2025. URL https://arxiv.org/abs/2502.07223.

Peng Qi, Xiaowen Lin, Leo Mehr, Zijian Wang, and Christopher D. Manning. Answering complex open-domain questions through iterative query generation. *ArXiv*, abs/1910.07000, 2019. URL https://api.semanticscholar.org/CorpusID:202773198.

Changle Qu, Sunhao Dai, Xiaochi Wei, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, Jun Xu, and Ji-Rong Wen. Towards completeness-oriented tool retrieval for large language models. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pp. 1930–1940. ACM, 2024. doi: 10.1145/3627673.3679847. URL https://arxiv.org/abs/2405.16089.

Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. Rocketqa: An optimized training approach to dense passage retrieval for open-domain question answering, 2021. URL http://arxiv.org/abs/2010.08191.

Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389, April 2009. ISSN 1554-0669. doi: 10.1561/1500000019. URL https://doi.org/10.1561/1500000019.

Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. In *Advances in Neural Information Processing Systems*, volume 31, 2018. URL https://arxiv.org/abs/1810.04650.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. In *Advances in Neural Information Processing Systems 36*, pp. 38154–38180, 2023. doi: 10.52202/075280-1657. URL https://doi.org/10.52202/075280-1657.

Shuzheng Si, Haozhe Zhao, Yu Lei, Qingyi Wang, Dingwei Chen, Zhitong Wang, Zhenhailong Wang, Kangyang Luo, Zheng Wang, Gang Chen, Fanchao Qi, Minjia Zhang, and Maosong Sun. From context to skills: Can language models learn from context skillfully? *ArXiv*, abs/2604.27660, 2026. URL https://api.semanticscholar.org/CorpusID:287915777.

Georgios Sidiropoulos and Evangelos Kanoulas. Improving the robustness of dense retrievers against typos via multi-positive contrastive learning. In *Advances in Information Retrieval: 46th European Conference on Information Retrieval (ECIR)*, 2024. URL https://arxiv.org/abs/2403.10939.

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. doi: 10.48550/arXiv.2604.24594. URL https://arxiv.org/abs/2604.24594.

Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. doi: 10.48550/arXiv.1807.03748. URL https://arxiv.org/abs/1807.03748.

Vladimir Vapnik and Akshay Vashist. 2009 special issue: A new learning paradigm: Learning using privileged information. *Neural Netw.*, 22(5–6):544–557, July 2009. ISSN 0893-6080. doi: 10.1016/j.neunet.2009.06.042. URL https://doi.org/10.1016/j.neunet.2009.06.042.

Benben Wang, Minghao Tang, Hengran Zhang, Jiafeng Guo, and Keping Bi. Training dense retrievers with multiple positive passages. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2026a. doi: 10.1145/3770855.3818053. URL https://arxiv.org/abs/2602.12727.

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. doi: 10.48550/arXiv.2305.16291. URL https://arxiv.org/abs/2305.16291.

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. doi: 10.48550/arXiv.2606.03565. URL https://arxiv.org/abs/2606.03565.

Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V. Le. Unsupervised data augmentation for consistency training. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=zeFrfgyZln.

Wenhan Xiong, Xiang Lorraine Li, Srini Iyer, Jingfei Du, Patrick Lewis, William Yang Wang, Yashar Mehdad, Wen tau Yih, Sebastian Riedel, Douwe Kiela, and Barlas Oğuz. Answering complex open-domain questions with multi-hop dense retrieval. 2020. URL http://arxiv.org/abs/2009.12756.

Yue Yao, Shengyuan Wang, Xin Chen, Minke Zhang, Jia He, Bingjun Luo, and Tom Gedeon. Skilltrace: Traversing a query-skill graph for composable llm agents. *arXiv preprint arXiv:2608.02356*, 2026. doi: 10.48550/arXiv.2608.02356. URL https://arxiv.org/abs/2608.02356.

HongChien Yu, Chenyan Xiong, and Jamie Callan. Improving query representations for dense retrieval with pseudo relevance feedback. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pp. 3592–3596. ACM, 2021. doi: 10.1145/3459637.3482124. URL https://doi.org/10.1145/3459637.3482124.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

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.

Chen Zhao, Chenyan Xiong, Jordan Boyd-Graber, and Hal Daumé III. Multi-step reasoning over unstructured text with beam dense retrieval. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4635–4641, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.368. URL https://aclanthology.org/2021.naacl-main.368/.

Yan-Zhao Zheng, Zhen-Tao Zhang, Chao Ma, Yuan-Qiang Yu, Ji-Huai Zhu, Yong-Liang Wu, Tianze Xu, Baohua Dong, Hangcheng Zhu, Ruo-Hui Huang, and Gang Yu. Skillrouter: Skill routing for llm agents at scale. *ArXiv*, 2026. URL https://arxiv.org/abs/2603.22455.

A Additional Experimental Details

A.1 Additional Benchmark Evaluation

We further evaluate SkillComp on SRA-Bench (Su et al., 2026) as an additional cross-domain analysis. Unlike the benchmarks used in the main experiments, SRA-Bench combines tasks from multiple source benchmarks spanning substantially different domains and skill types. This setting allows us to examine how SkillComp transfers beyond the skill distribution seen during training and how this transfer relates to the coverage of target skills in the training data. SkillComp is trained on 63,259 SkillRet queries over 10,123 skills. Among these skills, 62.3% belong to software engineering, while mathematics, logic, and medicine are not represented.

Table 5 compares the SRA-Bench skills with this training distribution. The 25,626 non-gold skills remain relatively close to the training skills, with a mean nearest-skill similarity of 0.886. In contrast, none of the 636 gold skills appears in the training data, and 76.7% are far from the training skills. Thus, the larger distribution shift lies in the skills that actually need to be retrieved rather than in the candidate corpus as a whole.

Table 6 further reports performance by domain. SkillComp obtains its largest gains over the backbone on ToolQA and BigCodeBench, whose gold skills are closest to the training distribution. CHAMP, whose gold skills are the most distant, is the only domain where SkillComp underperforms the backbone. LogicBench does not follow this trend, but it contains only 19 gold skills, making its domain-level result more sensitive to individual cases. These results suggest that SkillComp transfers more effectively when the target skill types are better covered by the training data. Although SkillComp can generalize to unseen skills, performance becomes less reliable as the required skills move further from the training distribution. Improving generalization to skills that lie further from the training distribution is therefore an important direction for future work.

Table 5: Similarity between SRA Bench skills and SkillComp training skills. Sim. is the mean cosine similarity between each gold skill and its nearest SkillRet training skill, computed with Qwen3-Embedding-0.6B over skill names and descriptions. Far is the percentage of gold skills with Sim. below 0.742, the 5th percentile for SkillRet test skills. Domains are sorted by Sim. The final row reports SRA-Bench non-gold skills as a reference; their mean Sim. is comparable to that of SkillRet test skills (0.886 vs. 0.874).

Domain	Skill type	#Queries	#Skills	Multi (%)	Sim.	Far (%)
ToolQA	Tool workflows	1,430	14	0.0	0.806	14.3
BigCodeBench	Software libraries	1,140	139	100.0	0.755	46.0
MedCalcBench	Medical calculators	1,100	55	0.0	0.706	72.7
LogicBench	Logic patterns	760	19	0.0	0.653	100.0
TheoremQA	Theorems	747	320	0.0	0.640	85.9
CHAMP	Math concepts	223	89	48.9	0.561	98.9
All	All types	5,400	636	23.1	0.664	76.7
Non-gold (reference)	Public agent skills	–	25,626	–	0.886	5.7

A.2 Co-required Skill Selection

For a co-required pair, neither skill has a directional dependency on the other, so the relation itself does not determine which skill should be appended. We therefore examine whether this choice is sensitive to the skill's current relevance to the plain query. SkillComp-HighScore selects the higher-scoring skill, while the final SkillComp selects the lower-scoring skill.

Table 7 compares the two choices. On SkillRet, selecting the lower-scoring skill gives slightly higher recall and completeness. On SkillBench, the differences are mixed and generally small. Overall, the results suggest that co-required expansion is not highly sensitive to which member of the pair is selected, although the lower-scoring choice performs slightly better overall.

Table 6: Retrieval results for on SRA Bench.

Domain	Model	Hit@1	C@10	R@10	NDCG@10	C@20	R@20	NDCG@20
TheoremQA (747)	Qwen3-Embedding-0.6B	68.41	88.76	88.76	78.81	91.57	91.57	79.54
	SkillComp-Embedding-0.6B	71.75	90.50	90.50	81.14	93.71	93.71	81.95
LogicBench (760)	Qwen3-Embedding-0.6B	3.03	7.63	7.63	4.86	11.71	11.71	5.90
	SkillComp-Embedding-0.6B	18.03	50.66	50.66	31.94	61.71	61.71	34.76
ToolQA (1,430)	Qwen3-Embedding-0.6B	0.00	1.40	1.40	0.54	7.62	7.62	2.04
	SkillComp-Embedding-0.6B	17.76	59.51	59.51	37.55	72.59	72.59	40.83
CHAMP (223)	Qwen3-Embedding-0.6B	24.22	42.15	54.80	38.23	56.50	69.91	42.70
	SkillComp-Embedding-0.6B	21.08	32.29	45.25	31.59	42.60	56.61	34.87
MedCalcBench (1,100)	Qwen3-Embedding-0.6B	47.64	71.82	71.82	59.20	78.27	78.27	60.82
	SkillComp-Embedding-0.6B	44.36	73.91	73.91	57.74	83.45	83.45	60.14
BigCodeBench (1,140)	Qwen3-Embedding-0.6B	34.47	11.49	35.81	31.69	16.93	43.84	34.43
	SkillComp-Embedding-0.6B	47.72	27.89	58.60	49.61	41.23	70.91	53.88
Overall (5,400)	Qwen3-Embedding-0.6B	27.87	32.52	38.17	32.06	38.19	44.42	33.79
	SkillComp-Embedding-0.6B	37.15	57.69	64.70	49.20	68.33	75.18	52.11

Table 7: **Effect of co-required skill selection.** SkillComp-HighScore appends the higher-scoring skill in each co-required pair, while the final SkillComp appends the lower-scoring skill. Results use plain-query full-corpus retrieval. All values are percentages.

		SkillRet				SkillBench			
Variant	Co-required selection	R@10	R@20	C@10	C@20	R@10	R@20	C@10	C@20
	Higher-scoring (stronger)	86.67	91.21	77.13	84.39	**71.67**	77.80	53.33	60.00
SkillComp	Lower-scoring (weaker)	**87.38**	**91.90**	**78.29**	**85.23**	70.50	**80.29**	**53.33**	**66.67**

A.3 Training objectives of ablation variants

Table 8 lists the per-task training objective of each variant in Table 3. Each task's loss averages the losses of the query views used for that task: with both a plain and an expanded view, each view loss receives weight 0.5; with only the plain view, the plain-query loss receives weight 1. Because every task in SkillComp and *Random Appended Skill* has both views, this averaging rescales Eq. (5) by a constant factor. In the two variants without single-skill expansion, single-skill tasks use only the plain view, whereas multi-skill tasks split their weight between the two views. The comparisons for complete-set supervision (*w/o Appended Supervision* vs. *w/o Single-Skill Expansion*) and relation-guided selection (*Random Appended Skill* vs. SkillComp) keep the objective weights fixed and therefore isolate the corresponding design choice. The comparison between *w/o Single-Skill Expansion* and SkillComp changes both the single-skill objective and the relative weight of single-skill tasks.

Table 8: Per-task training objectives of the ablation variants in Table 3. $\mathcal{A}_q$ follows the relation-based rule in §4.1, except in *Random Appended Skill*, where a single r is sampled uniformly from G for each task.

Variant	Single-skill task	Multi-skill task
Base	$\mathcal{L}_D$	$\mathcal{L}_D$
w/o Appended Supervision	$\mathcal{L}_D$	$0.5\,\mathcal{L}_D + 0.5\,\frac{1}{\lvert\mathcal{A}_q\rvert}\sum_{A\in\mathcal{A}_q}\mathcal{L}_{\text{set}}(q_A; G\setminus A)$
w/o Single-Skill Expansion	$\mathcal{L}_D$	$0.5\,\mathcal{L}_D + 0.5\,\frac{1}{\lvert\mathcal{A}_q\rvert}\sum_{A\in\mathcal{A}_q}\mathcal{L}_{\text{set}}(q_A; G)$
Random Appended Skill	$0.5\,\mathcal{L}_D + 0.5\,\mathcal{L}_{\text{set}}(q_s; G)$	$0.5\,\mathcal{L}_D + 0.5\,\mathcal{L}_{\text{set}}(q_r; G)$
SkillComp	$0.5\,\mathcal{L}_D + 0.5\,\mathcal{L}_{\text{set}}(q_s; G)$	$0.5\,\mathcal{L}_D + 0.5\,\frac{1}{\lvert\mathcal{A}_q\rvert}\sum_{A\in\mathcal{A}_q}\mathcal{L}_{\text{set}}(q_A; G)$

A.4 Implementation Details

We initialize from Qwen3-Embedding-0.6B and Qwen3-Embedding-8B and use the same two-stage training procedure for both models. Stage I fine-tunes all model parameters with the standard skill

retrieval objective. Stage II starts from the resulting checkpoint, freezes the backbone, and trains LoRA adapters with the SkillComp objective.

Hard negative construction. The same selected skill r may co-occur with different required skills across tasks, so a skill required alongside r in one task is not necessarily relevant to another. To help the retriever distinguish these cases, we construct hard negatives using other training tasks that contain the same r. We collect skills that co-occur with r in those tasks but are not included in the current required set G. These skills are plausible given r but are not required for the current query, so they provide informative hard negatives. If fewer than eight such candidates are available, we fill the remaining slots with the highest-ranked non-required skills from dense retrieval.

Stage I: Fine-tuning for skill-domain retrieval. We first fine-tune all model parameters with the standard skill retrieval objective $\mathcal{L}_{\text{set}}(q; G_q)$. Pretrained embedding models are designed for general semantic matching, whereas skill retrieval requires matching a task to the executable capabilities needed to complete it. Stage I therefore adapts the model to task–skill relevance through direct training on $q \to G_q$. The resulting checkpoint is used to initialize Stage II.

Stage II: LoRA training. We then freeze the checkpoint-20 backbone and train separate LoRA adapters for the query and skill encoders, initialized from the same state. Expanded queries constructed from the annotated skill relations are introduced only at this stage, and each variant is trained with its corresponding SkillComp objective. The plain query q and expanded query q_r use the same query encoder, so their losses update the same query-side parameters. This setup allows all SkillComp variants to start from the same retrieval-adapted checkpoint, making their differences attributable to the Stage-II training design.

Table 9 compares the Stage-I checkpoint with the final Stage-II model under plain-query inference. Stage II provides further gains over Stage I, showing that SkillComp training improves retrieval beyond the initial adaptation to the skill domain.

Training Configuration. Tables 10 and 11 report the optimizer settings, learning rates, batch sizes, LoRA configurations, and training budgets.

Table 9: Retrieval performance of Stage I and Stage II. Stage I provides the initialization for Stage-II SkillComp training, and SkillComp denotes the final Stage-II model.

		SkillRet					SkillBench				
Scale	Checkpoint	Hit@1	R@10	R@20	C@10	C@20	Hit@1	R@10	R@20	C@10	C@20
0.6B	Stage I	77.03	83.12	87.99	71.00	78.23	65.33	68.19	74.33	50.67	58.67
0.6B	SkillComp	79.25	87.38	91.90	78.29	85.23	72.00	70.50	80.29	53.33	66.67
8B	Stage I	78.05	87.46	92.07	78.33	85.59	57.33	68.70	75.73	54.67	60.00
8B	SkillComp	84.23	92.53	95.54	86.45	91.57	66.67	72.32	80.26	54.67	64.00

Table 10: Stage-I retrieval-adaptation configuration.

Configuration	Qwen3-Embedding-0.6B	Qwen3-Embedding-8B
Trainable parameters	General encoder	General encoder
Objective	$\mathcal{L}_D$	$\mathcal{L}_D$
Optimizer	AdamW	AdamW
Learning rate	2×10^{-5}	2×10^{-5}
Weight decay	0.01	0.01
LR warmup	10%	10%
Temperature	0.05	0.05
Maximum sequence length	8,192	8,192
Hard negatives	in-batch	in-batch
GPUs / batch per GPU	4/96	4/20
Gradient accumulation	1	1
Effective query batch	384	80
Training duration	15% Training data	15% Training data

Table 11: Stage-II dual-LoRA SkillComp configuration.

Configuration	Qwen3-Embedding-0.6B	Qwen3-Embedding-8B
Backbone	Frozen Stage-I backbone	Frozen Stage-I backbone
Trainable parameters	Query/Skill LoRA	Query/Skill LoRA
Target projections	Q/K/V/O, all 28 layers	Q/K/V/O, all 28 layers
LoRA rank / scaling	$8/16$	$8/16$
LoRA dropout	0	0
Objective	Joint SkillComp	Joint SkillComp
Optimizer	AdamW	AdamW
Learning rate	5×10^{-5}	5×10^{-5}
Weight decay	0.01	0.01
Warmup / gradient clip	10 steps / 1.0	10 steps / 1.0
Temperature / seed	$0.05/42$	$0.05/42$
Hard negatives per query	8	8
GPUs / logical batch per GPU	$2/96$	$2/96$
Gradient accumulation	2	2
Effective query batch	384	384
Training duration	2 epochs / 330 steps	2 epochs / 330 steps
Total Stage-I + Stage-II time	16 hours	20 hours

B Local Gradient Analysis

Setup. We measure per-task gradients at step 50 of SkillComp training on 768 two-skill training tasks, pooled from two disjoint batches of 384 tasks. Each two-skill task has a single expanded query q_r; r denotes the appended skill and s the other required skill. Let $g_t = \nabla_\theta \ell(q, t; \mathcal{N}_q)$ and $h_t = \nabla_\theta \ell(q_r, t; \mathcal{N}_q)$ for $t \in \{r, s\}$, so that $g_D = (g_r + g_s)/2$, the Partial expanded-view gradient is $h_C = h_s$, and the complete-set gradient is $h_G = (h_r + h_s)/2$. Both views share a fixed candidate pool of 3,114 in-batch and hard-negative skills as a surrogate for the training negatives. Per-task gradients are taken over the query-encoder LoRA parameters, and all four directions are evaluated at the same parameter point.

Results. Figure 6 reports four views of the same measurements. (a) The base gradient aligns unevenly with the two targets, as described by Equation (2): the median of $\cos(g_t, g_D)$ is 0.21 for r and 0.78 for s, and the cosine is positive for 479 and 663 tasks, respectively. (b) The same-target pairs (g_r, h_r) and (g_s, h_s) are positively aligned for all 768 tasks, with medians of 0.64 and 0.81. The cross-target pairs (g_r, h_s) and (g_s, h_r) are negative for 749 and 741 tasks. (c) The joint direction gives larger first-order descent than the base direction, $\langle g_D, g_D + h_G \rangle > \|g_D\|^2$, for 765 tasks. Equivalently, $\langle g_D, h_G \rangle > 0$, the aggregate form of Equation (3). (d) Complete-set supervision gives r larger first-order descent than Partial supervision, $\langle g_r, h_G \rangle > \langle g_r, h_C \rangle$, for 764 tasks. For two skills this is equivalent to $\langle g_r, h_r \rangle > \langle g_r, h_C \rangle$, the condition of Proposition 1. This difference splits as $\langle g_r, h_G \rangle - \langle g_r, h_C \rangle = \frac{1}{2}\langle g_r, h_r \rangle - \frac{1}{2}\langle g_r, h_s \rangle$: the added-target term is positive for all 768 tasks and the reweighting term for 749. Under per-target weighting, the comparison reduces to the first term (Equation (33)). The Partial direction conflicts with g_r in 749 tasks, whereas the complete-set direction is positively aligned with g_r in 377 tasks. The evidence therefore supports a relative advantage of complete-set supervision for r, not positive transfer to r in every task.

C SkillComp Training Algorithm

Notation. θ denotes the backbone parameters. In Stage I, the query and skill encoders share this backbone, so $\theta_q = \theta_s = \theta$ in the notation of Section 2. $\theta^\star$ denotes the resulting backbone, which is frozen and shared by both encoders in Stage II, and $\phi = (\phi_q, \phi_s)$ the query and skill LoRA parameters trained on top of it. For task i, $\mathcal{H}_i$ is its hard-negative cache and $\mathcal{A}_i$ its appended skill sets. $\mathrm{Refresh}$ updates these sets using the current retriever and eight hard negatives per task (Section 5; Appendix F). Init initializes adapters; Update applies an optimizer step. Losses use $\mathcal{N}_i$ and temperature τ as in Equation (1).

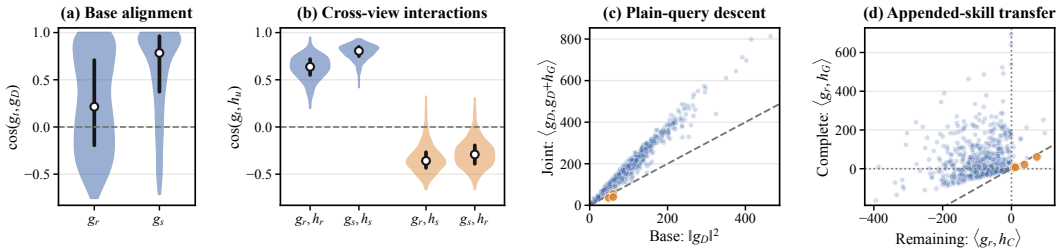


Figure 6: **Local gradient interactions under complete-set supervision.** (a) Alignment of each plain-query target gradient with the base gradient g_D. (b) Same-target and cross-target cross-view alignment; violins show densities with medians and interquartile ranges. (c) First-order descent of the aggregate plain-query loss under the base direction g_D and the joint direction $g_D + h_G$. (d) First-order plain-query transfer to r under Partial (h_C) and complete-set (h_G) supervision. Dashed diagonals mark equality; orange points lie on or below it.

Algorithm 1 Two-stage SkillComp training

Require: $\mathcal{D} = \{(q_i, G_i)\}$, skill collection $\mathcal{S}$, initial θ, hard-negative caches $\{\mathcal{H}_i\}$, steps T_1, T_2
Ensure: Adapted backbone $\theta^\star$ and query/skill adapters ϕ

Stage I: Plain-query training

for $t = 0, \ldots, T_1 - 1$ **do**
  $\mathcal{B} \sim \mathcal{D}$; $\mathcal{C}_\mathcal{B} \leftarrow \bigcup_{j\in\mathcal{B}} G_j$ ▷ In-batch candidates
  $\mathcal{N}_i \leftarrow (\mathcal{C}_\mathcal{B} \cup \mathcal{H}_i) \setminus G_i$ for each $i \in \mathcal{B}$
  $\mathcal{L}_1 \leftarrow |\mathcal{B}|^{-1} \sum_{i\in\mathcal{B}} \mathcal{L}_D(q_i; G_i)$
  $\theta \leftarrow \mathrm{Update}(\theta, \nabla_\theta \mathcal{L}_1)$
end for
$\theta^\star \leftarrow \theta$; $\phi \leftarrow \mathrm{Init}()$

Stage II: Joint plain/expanded-query training

for $t = 0, \ldots, T_2 - 1$ **do**
  if $t \bmod 50 = 0$ **then**
    $\{\mathcal{A}_i, \mathcal{H}_i\}_{i\in\mathcal{D}} \leftarrow \mathrm{Refresh}(\theta^\star, \phi, \mathcal{D}, \mathcal{S})$
  end if
  $\mathcal{B} \sim \mathcal{D}$; $\mathcal{C}_\mathcal{B} \leftarrow \bigcup_{j\in\mathcal{B}} G_j$
  $\mathcal{N}_i \leftarrow (\mathcal{C}_\mathcal{B} \cup \mathcal{H}_i) \setminus G_i$ for each $i \in \mathcal{B}$ ▷ Shared across views
  $\mathcal{L}_2 \leftarrow \frac{1}{|\mathcal{B}|} \sum_{i\in\mathcal{B}} \left[\mathcal{L}_D(q_i; G_i) + \frac{1}{|\mathcal{A}_i|} \sum_{A\in\mathcal{A}_i} \mathcal{L}_{\mathrm{set}}(q_i \oplus A; G_i) \right]$
  $\phi \leftarrow \mathrm{Update}(\phi, \nabla_\phi \mathcal{L}_2)$
end for
return $\theta^\star, \phi$

D Evaluation Metrics

Let $\mathcal{Q}$ denote the evaluation queries. For a query $q \in \mathcal{Q}$ with required skill set G_q, let $P_K(q)$ denote the Top-K skills retrieved with q.

Retrieval metrics. Recall and Completeness are computed per query and averaged over queries:

$$R@K = \frac{1}{|\mathcal{Q}|} \sum_{q\in\mathcal{Q}} \frac{|G_q \cap P_K(q)|}{|G_q|}, \qquad C@K = \frac{1}{|\mathcal{Q}|} \sum_{q\in\mathcal{Q}} \mathbf{1}[G_q \subseteq P_K(q)]\,. \tag{6}$$

$R@K$ measures the fraction of required skills retrieved, and $C@K$ measures the fraction of queries whose complete required set is retrieved. Hit@1 is the fraction of queries with $P_1(q) \cap G_q \neq \emptyset$. MRR@10 averages the reciprocal rank of the first required skill in $P_{10}(q)$ and assigns zero when $P_{10}(q) \cap G_q = \emptyset$.

Gold-skill expansion. For the diagnostic in Figure 4, $\mathcal{Q}$ is the SkillRet multi-skill subset, and r_q is the highest-ranked required skill in SkillComp's plain-query ranking, shared across models. For each

model, let $P = P_{10}(q)$ and $P' = P_{10}(q \oplus r_q)$. Excluding r_q, the previously missed and previously retrieved required skills are

$$M_q = (G_q \setminus P) \setminus \{r_q\}, \qquad A_q = (G_q \cap P) \setminus \{r_q\}. \tag{7}$$

Found and Lost are averaged over the queries for which the corresponding set is nonempty:

$$\text{Found} = \frac{1}{|\mathcal{Q}_M|} \sum_{q \in \mathcal{Q}_M} \frac{|M_q \cap P'|}{|M_q|}, \qquad \text{Lost} = \frac{1}{|\mathcal{Q}_A|} \sum_{q \in \mathcal{Q}_A} \frac{|A_q \setminus P'|}{|A_q|}, \tag{8}$$

where $\mathcal{Q}_M = \{q \in \mathcal{Q} : M_q \neq \emptyset\}$ and $\mathcal{Q}_A = \{q \in \mathcal{Q} : A_q \neq \emptyset\}$. Found measures whether the appended skill helps recover other missed required skills, whereas Lost measures whether it displaces required skills already retrieved by the plain query. Because the two quantities have different denominators, they are not combined into a net gain.

E Derivations for Multi-Skill Optimization and Expanded-Query Supervision

This appendix provides the derivations underlying the gradient analysis in Sections 3 and 4. Following the main text, we derive the base gradient interactions, the effect of auxiliary supervision, and the role of the expanded-view target set. We then connect these results to the averaged SkillComp objective and the gradient evaluation in Appendix B.

Setup and parameter convention. Fix a query q, a finite nonempty required set G, a finite nonempty negative set $\mathcal{N}_q$ disjoint from G, and a temperature $\tau > 0$. All query views use this same negative set. The appended sets $\mathcal{A}_q$ and their expanded queries are held fixed when taking derivatives and comparing updates. This fixes candidate identities; their encoder representations remain differentiable functions of the trainable parameters. We use $\theta = (\theta_q, \theta_s)$ for the trainable coordinates of the query and skill encoders, with frozen parameters held constant. In Stage I the two encoders share one backbone, so θ is that single parameter vector and each gradient collects contributions through both encoders. In Stage II these coordinates are the two LoRA adapters, denoted by $\phi = (\phi_q, \phi_s)$ in Algorithm 1; the frozen backbone is not part of the update vector. Each target loss is assumed differentiable at the current parameter point. All gradients are evaluated at that point, and each $o(\eta)$ is taken as $\eta \to 0^+$ with the task, selected views, candidate set, and update directions fixed. The comparisons concern Euclidean gradient steps at a common step size.

E.1 Base Gradient Interactions

Base objective and shared-gradient interaction. Let G be the complete required set for query q, with $m = |G|$. Following Equation (1), define the base target loss and its gradient for each $s \in G$ as

$$\ell_s(\theta) := \ell(q, s; \mathcal{N}_q), \qquad g_s = \nabla_\theta \ell_s(\theta). \tag{9}$$

The base set loss is

$$\mathcal{L}_D(q; G) := \frac{1}{m} \sum_{t \in G} \ell_t(\theta), \qquad g_D = \nabla_\theta \mathcal{L}_D(q; G) = \frac{1}{m} \sum_{t \in G} g_t. \tag{10}$$

For a gradient step $\theta^+ = \theta - \eta g_D$, let ℓ_s^+ denote the target loss after the update. First-order Taylor expansion gives

$$\begin{aligned} \ell_s^+ - \ell_s(\theta) &= -\eta \langle g_s, g_D \rangle + o(\eta) \\ &= -\frac{\eta}{m} \left[\|g_s\|^2 + \sum_{t \in G \setminus \{s\}} \langle g_s, g_t \rangle \right] + o(\eta). \end{aligned} \tag{11}$$

The self-term $\|g_s\|^2$ supports descent for ℓ_s, whereas the cross-target terms $\langle g_s, g_t \rangle$ can reinforce or oppose this update. The base objective therefore supervises every required skill, with progress on each skill determined by both its self-term and its cross-target interactions. This component-gradient view is standard in analyses of jointly optimized objectives (Yu et al., 2020; Liu et al., 2021).

E.2 Auxiliary Transfer and Uneven Optimization

Auxiliary transfer relative to the base update. Fix the query and negative set, and evaluate gradients at the current parameter point θ. Let H be a fixed auxiliary gradient, and define the base-only and augmented updates as

$$\theta_B = \theta - \eta g_D, \qquad \theta_A = \theta - \eta(g_D + H). \tag{12}$$

When ℓ_s is differentiable at θ, Taylor expansion gives

$$\begin{aligned} \ell_s(\theta_B) &= \ell_s(\theta) - \eta\langle g_s, g_D\rangle + o(\eta), \\ \ell_s(\theta_A) &= \ell_s(\theta) - \eta\langle g_s, g_D + H\rangle + o(\eta). \end{aligned} \tag{13}$$

Subtracting yields

$$\Delta_s(H) := \ell_s(\theta_B) - \ell_s(\theta_A) = \eta\langle g_s, H\rangle + o(\eta), \tag{14}$$

which proves Equation (3).

For an appended set $A \subseteq G$, let $q_A = q \oplus A$ and define the expanded-query same-target gradient

$$h_s^{(A)} = \nabla_\theta \ell(q_A, s; \mathcal{N}_q). \tag{15}$$

We write $h_s^{(r)}$ when $A = \{r\}$. To establish the positive local transfer condition stated in Section 3, choose $H = h_s^{(A)}$ in Equation (14). If $\langle g_s, h_s^{(A)}\rangle > 0$, its positive linear coefficient dominates the $o(\eta)$ remainder for sufficiently small $\eta > 0$. Hence $\Delta_s(h_s^{(A)}) > 0$, or equivalently

$$\ell_s\Big(\theta - \eta(g_D + h_s^{(A)})\Big) < \ell_s(\theta - \eta g_D). \tag{16}$$

This establishes additional local progress relative to the base-only update.

Relative improvement and descent. Writing $p_s = \langle g_s, g_D\rangle$ and $\delta_s(H) = \langle g_s, H\rangle$, the augmented update satisfies

$$\ell_s(\theta_A) - \ell_s(\theta) = -\eta\big[p_s + \delta_s(H)\big] + o(\eta). \tag{17}$$

Thus $\delta_s(H) > 0$ gives a lower loss relative to the base update, while $p_s + \delta_s(H) > 0$ gives descent relative to the current loss, in each case for sufficiently small $\eta > 0$. A zero first-order coefficient leaves the sign to higher-order terms. Because $g_D + H$ generally has a larger norm than g_D, these comparisons characterize the sign of the auxiliary interaction at a common step size rather than a gain per unit step length; by Cauchy–Schwarz, no unit direction decreases $\mathcal{L}_D$ faster to first order than $g_D/\|g_D\|$.

Disparity in first-order progress. Section 3 uses p_s to quantify the base descent coefficient. For two skills w, v with $p_w < p_v$, let $D = p_v - p_w > 0$ and $d = \delta_w(H) - \delta_v(H)$. Their augmented coefficients $p_t^{\mathrm{aug}} = p_t + \delta_t(H)$ obey the exact identity

$$p_v^{\mathrm{aug}} - p_w^{\mathrm{aug}} = D - d. \tag{18}$$

If $0 < d \le D$, the gap decreases from D to a value in $[0, D)$ without reversing the order. If also $\delta_w(H) > 0$, the slower skill receives positive auxiliary transfer. These conditions make explicit why improving one skill alone does not establish reduced disparity across skills.

Loss-gap comparison. Loss values provide a separate comparison from the descent coefficients: a higher current loss need not have a smaller p_s. Let $w, v \in G$ satisfy $\ell_w(\theta) > \ell_v(\theta)$, so w is currently less well matched under the plain query. For the same fixed auxiliary direction H, Equation (14) gives

$$\Delta_w(H) - \Delta_v(H) = \eta\langle g_w - g_v, H\rangle + o(\eta). \tag{19}$$

If

$$\langle g_w, H\rangle > \max\{0, \langle g_v, H\rangle\}, \tag{20}$$

then, for sufficiently small $\eta > 0$, $\Delta_w(H) > 0$ and $\Delta_w(H) > \Delta_v(H)$. Moreover, differentiability and the strict initial ordering ensure that w remains the higher-loss skill after either sufficiently small update. Using the same signed ordering, the augmented loss gap satisfies

$$\begin{aligned} \big[\ell_w(\theta_A) - \ell_v(\theta_A)\big] &= \big[\ell_w(\theta_B) - \ell_v(\theta_B)\big] - \big[\Delta_w(H) - \Delta_v(H)\big] \\ &< \ell_w(\theta_B) - \ell_v(\theta_B). \end{aligned} \tag{21}$$

Equation (20) therefore gives a sufficient local condition for improving the higher-loss skill and narrowing the loss gap relative to the base-only update.

E.3 EXPANDED-VIEW TARGET SUPPORT

General target-set decomposition. For a fixed appended set A, define $h_t^{(A)}$ for every $t \in G$ as in Equation (15). For any nonempty $T \subseteq G$, linearity of differentiation and of the inner product gives

$$h_T^{(A)} = \nabla_\theta \mathcal{L}_{\text{set}}(q_A; T) = \frac{1}{|T|} \sum_{t \in T} h_t^{(A)},$$

$$\langle g_s, h_T^{(A)} \rangle = \frac{1}{|T|} \left[\underbrace{\mathbf{1}[s \in T] \langle g_s, h_s^{(A)} \rangle}_{\text{same-target}} + \underbrace{\sum_{t \in T \setminus \{s\}} \langle g_s, h_t^{(A)} \rangle}_{\text{cross-target}} \right]. \tag{22}$$

Here the negative set remains $\mathcal{N}_q$, excluding all of G even when T is a proper subset. This proves Equation (4). The choice $T = G$ retains a same-target term for every required skill; the sign of its net auxiliary effect is determined by the entire sum.

Remaining-target gradient decomposition. For a proper subset $A \subsetneq G$, as for every multi-skill task, let $C_A = G \setminus A$ be the nonempty remaining target set. We call $T = C_A$ *Partial* or remaining-target supervision, and $T = G$ *Full* or complete-set supervision. Both retain the same plain-query objective $\mathcal{L}_D(q; G)$. Dropping the superscript (A) for a fixed appended set, define

$$h_C = \frac{1}{|C_A|} \sum_{t \in C_A} h_t. \tag{23}$$

For any $s \in C_A$, its interaction with the remaining-target gradient decomposes exactly as

$$\begin{aligned} \langle g_s, h_C \rangle &= \frac{1}{|C_A|} \sum_{t \in C_A} \langle g_s, h_t \rangle \\ &= \frac{1}{|C_A|} \left[\langle g_s, h_s \rangle + \sum_{\substack{t \in C_A \\ t \neq s}} \langle g_s, h_t \rangle \right]. \end{aligned} \tag{24}$$

The first term is the direct same-target interaction. The sum determines its net effect together with the cross-target interactions.

To compare Partial with the base update, Taylor expansion gives

$$\ell_s|_{\theta - \eta(g_D + h_C)} - \ell_s|_{\theta - \eta g_D} = -\eta \langle g_s, h_C \rangle + o(\eta). \tag{25}$$

A sufficient condition for Partial to yield a lower base target loss for s than the base update alone is

$$\langle g_s, h_s \rangle + \sum_{\substack{t \in C_A \\ t \neq s}} \langle g_s, h_t \rangle > 0. \tag{26}$$

Under this condition, the comparison holds for sufficiently small $\eta > 0$. As shown in Equation (42), the lower target loss corresponds to a larger loss-aligned relative score $R_\tau(q, s)$.

Appended skills under Partial supervision. Because $C_A = G \setminus A$, the auxiliary part of the Partial objective affects each appended skill $r \in A$ through the remaining-target gradient alone:

$$\langle g_r, h_C \rangle = \frac{1}{|C_A|} \sum_{t \in C_A} \langle g_r, h_t \rangle. \tag{27}$$

Every term is cross-target because $t \notin A$, so its sign is unconstrained. For example, for $g_r \neq 0$, taking $h_t = -g_r$ for every $t \in C_A$ gives $\langle g_r, h_C \rangle = -\|g_r\|^2 < 0$. Adding A to the expanded-query target set restores the direct same-target contribution of each appended skill.

Proposition 1 (Complete-set versus remaining-target supervision)**.** *Let $m \geq 2$, let $A \subsetneq G$ be an appended set with $r \in A$, and let $h_A = \frac{1}{|A|}\sum_{t\in A} h_t$. Let $h_G = \frac{1}{m}\left(\sum_{t\in A} h_t + \sum_{t\in C_A} h_t\right) = \frac{|A|}{m}h_A + \frac{m-|A|}{m}h_C$ denote the complete-set gradient. Let ℓ_r^{Full} and $\ell_r^{\mathrm{Partial}}$ denote the plain-query target loss for r after the updates $\theta - \eta(g_D + h_G)$ and $\theta - \eta(g_D + h_C)$, respectively. Their first-order difference is*

$$\ell_r^{\mathrm{Full}} - \ell_r^{\mathrm{Partial}} = -\frac{\eta|A|}{m}\left(\langle g_r, h_A\rangle - \langle g_r, h_C\rangle\right) + o(\eta). \tag{28}$$

If $\langle g_r, h_A\rangle > \langle g_r, h_C\rangle$, the complete expanded-query objective yields a lower plain-query target loss for r than the Partial objective after these updates, for sufficiently small $\eta > 0$. For $A = \{r\}$, $h_A = h_r$.

Proof of Proposition 1. Complete-set supervision extends the expanded-query target from C_A to $G = C_A \cup A$. The two expanded-query gradients obey

$$\begin{aligned} h_G &= \frac{1}{m}\left(\sum_{t\in A} h_t + \sum_{t\in C_A} h_t\right) \\ &= \frac{|A|}{m}h_A + \frac{m-|A|}{m}h_C, \end{aligned} \tag{29}$$

and therefore

$$h_G - h_C = \frac{|A|}{m}(h_A - h_C). \tag{30}$$

Extending the mean set loss from C_A to G both introduces the appended-skill gradients and rescales the existing remaining-target gradient. Taylor expansion around θ then gives

$$\begin{aligned} \ell_r^{\mathrm{Full}} - \ell_r^{\mathrm{Partial}} &= -\eta\langle g_r, h_G - h_C\rangle + o(\eta) \\ &= -\frac{\eta|A|}{m}\left(\langle g_r, h_A\rangle - \langle g_r, h_C\rangle\right) + o(\eta). \end{aligned} \tag{31}$$

If $\langle g_r, h_A\rangle > \langle g_r, h_C\rangle$, the linear coefficient is strictly negative and dominates the remainder for sufficiently small $\eta > 0$, proving the result. By Equation (42), the same comparison corresponds to a larger $R_\tau(q, r)$.

Target addition versus reweighting. Because both objectives average over their targets,

$$\langle g_r, h_G - h_C\rangle = \underbrace{\tfrac{1}{m}\textstyle\sum_{t\in A}\langle g_r, h_t\rangle}_{\text{added targets}} + \underbrace{\left(-\tfrac{|A|}{m}\langle g_r, h_C\rangle\right)}_{\text{reweighting}} \tag{32}$$

combines added targets with a reweighting of the remaining targets. To isolate the added targets, weight each remaining target by $1/m$ as in the complete objective, $h_C^\Sigma = \frac{1}{m}\sum_{t\in C_A} h_t = \frac{m-|A|}{m}h_C$. Then $h_G - h_C^\Sigma = \frac{1}{m}\sum_{t\in A} h_t$, and

$$\ell_r\big(\theta - \eta(g_D + h_G)\big) - \ell_r\big(\theta - \eta(g_D + h_C^\Sigma)\big) = -\frac{\eta}{m}\sum_{t\in A}\langle g_r, h_t\rangle + o(\eta). \tag{33}$$

For $A = \{r\}$, the same-target condition $\langle g_r, h_r\rangle > 0$ alone suffices; for $A = \{r, r'\}$, the sum also contains the cross-target term $\langle g_r, h_{r'}\rangle$.

Interpretation of the comparison. Proposition 1 compares Full with Partial at the same parameter point. A sufficient condition for positive local transfer from Full relative to the base update is

$$\langle g_r, h_G\rangle = \frac{|A|\langle g_r, h_A\rangle + (m-|A|)\langle g_r, h_C\rangle}{m} > 0. \tag{34}$$

Full can therefore improve on Partial while its auxiliary direction still opposes g_r. For $m = 2$ and $A = \{r\}$, the scalar gradients $g_r = h_r = 1$ and $g_s = h_s = -3$ give $g_D = h_G = -1$ and $h_C = -3$. Full has a more favorable first-order effect on r than Partial, but its auxiliary transfer to r is negative. These gradients are realized at $\theta = 0$ by identical query views with losses $\ell_r = \log(1 + e^{2\theta})$ and $\ell_s = \log(1 + e^{-6\theta})$, using one negative of score zero and $\tau = 1$. This example distinguishes the relative comparison in the proposition from positive transfer and from descent of the target loss.

E.4 The Averaged SkillComp Objective

Multiple expanded queries. Restoring the superscript (A), the auxiliary gradient of Equation (5) is

$$H_{\mathrm{SkillComp}} := \frac{1}{|\mathcal{A}_q|} \sum_{A \in \mathcal{A}_q} h_G^{(A)}, \qquad \nabla_\theta \mathcal{L}_{\mathrm{SkillComp}}(q; G) = g_D + H_{\mathrm{SkillComp}}. \tag{35}$$

Substitution into Equation (14) gives

$$\Delta_s(H_{\mathrm{SkillComp}}) = \frac{\eta}{|\mathcal{A}_q|} \sum_{A \in \mathcal{A}_q} \langle g_s, h_G^{(A)} \rangle + o(\eta). \tag{36}$$

The average inner product determines the local effect on s. An individual expanded query can have positive transfer while the average does not. Applying Equation (22) to each summand shows that complete-set supervision retains a same-target contribution for every required skill in every expanded query.

Single-skill tasks. When $G = \{s\}$, $\mathcal{A}_q = \{\{s\}\}$, $g_D = g_s$, and $H_{\mathrm{SkillComp}} = h_s^{(s)}$. Equation (36) reduces to $\Delta_s = \eta \langle g_s, h_s^{(s)} \rangle + o(\eta)$. There is no cross-target term. A strictly positive inner product gives positive local transfer and, since $\|g_s\|^2 + \langle g_s, h_s^{(s)} \rangle > 0$, descent of the plain-query loss for sufficiently small steps. The Partial comparison is defined only for $m \geq 2$, where C_A is nonempty.

Batch updates. For a fixed batch $\mathcal{B}$ of b tasks, write $g_{D,i}$ and $H_{\mathrm{SkillComp},i}$ for the corresponding per-task gradients, and let

$$\overline{g}_D = \frac{1}{b} \sum_{i \in \mathcal{B}} g_{D,i}, \qquad \overline{H} = \frac{1}{b} \sum_{i \in \mathcal{B}} H_{\mathrm{SkillComp},i}. \tag{37}$$

The gradient of the batch-mean plain-query loss $\overline{\mathcal{L}}_D = b^{-1} \sum_i \mathcal{L}_D(q_i; G_i)$ is $\overline{g}_D$. Applying the same Taylor argument gives

$$\overline{\mathcal{L}}_D(\theta - \eta \overline{g}_D) - \overline{\mathcal{L}}_D(\theta - \eta(\overline{g}_D + \overline{H})) = \eta \langle \overline{g}_D, \overline{H} \rangle + o(\eta). \tag{38}$$

For a target s of task i, the analogous coefficient is $\langle g_{i,s}, \overline{H} \rangle = b^{-1} \sum_j \langle g_{i,s}, H_{\mathrm{SkillComp},j} \rangle$. It includes interactions with other tasks. Consequently, the batch-level inner product describes aggregate transfer and need not have the same sign as each per-task or per-target interaction.

E.5 Connection to the Gradient Evaluation

Parameter blocks and gradient alignment. The per-task measurements in Figure 3 and Figure 6 evaluate the query-encoder LoRA block. To distinguish this restriction from the full trainable vector, write $\widetilde{g}_t = \nabla_{\theta_q} \ell(q, t; \mathcal{N}_q)$ and $\widetilde{h}_t = \nabla_{\theta_q} \ell(q_r, t; \mathcal{N}_q)$, where q_r is the single expanded query of each two-skill task. The figures and Appendix B abbreviate these restricted gradients as g_t, h_t. All decompositions above apply to this block by holding the skill-encoder coordinates fixed. For nonzero gradients, cosine and inner product have the same sign; the magnitude of a first-order loss change additionally depends on both gradient norms.

Quantities evaluated in the four panels. For the two-skill tasks, let $\widetilde{g}_D = (\widetilde{g}_r + \widetilde{g}_s)/2$, $\widetilde{h}_C = \widetilde{h}_s$, and $\widetilde{h}_G = (\widetilde{h}_r + \widetilde{h}_s)/2$. Figure 6(a) measures the directional component of base progress through $\cos(\widetilde{g}_t, \widetilde{g}_D)$. Panel (b) measures the same-target and cross-target components of Equation (22). Panel (c) compares the aggregate plain-query descent coefficients; their difference is exactly $\langle \widetilde{g}_D, \widetilde{h}_G \rangle$. Panel (d) compares the appended skill's auxiliary coefficients, with

$$\langle \widetilde{g}_r, \widetilde{h}_G \rangle - \langle \widetilde{g}_r, \widetilde{h}_C \rangle = \frac{1}{2} \left(\langle \widetilde{g}_r, \widetilde{h}_r \rangle - \langle \widetilde{g}_r, \widetilde{h}_C \rangle \right). \tag{39}$$

Thus points above equality satisfy the query-block version of Proposition 1. The reported 764 of 768 tasks above equality measure this relative advantage; the 377 tasks with $\langle \widetilde{g}_r, \widetilde{h}_G \rangle > 0$ measure positive auxiliary transfer over the base direction. They answer different comparisons.

Batch measurements and scope. The separate batch-level measurements in Appendix B include both encoder LoRA blocks. Writing their averaged directions as $\overline{g}_D$ and $\overline{h}_G$ distinguishes them from the per-task query-block quantities. Their positive cosines have the aggregate interpretation in Equation (38), for the expanded queries and candidate pools used in those measurements. For two-skill tasks, $\mathcal{A}_q = \{\{r\}\}$, so the per-task panels evaluate the complete auxiliary gradient of Equation (36), at one checkpoint. These measurements evaluate local gradient compatibility. Finite AdamW updates and retrieval rankings require their own evaluation, as reported for retrieval in Section 5.

E.6 Relation to the Contrastive Retrieval Score

Loss-aligned retrieval diagnostic. For a fixed negative set, the per-target loss comparisons above can also be expressed through a relative retrieval score. For $x \in \{q\} \cup \{q_A : A \in \mathcal{A}_q\}$, let

$$S_\theta(x, s) = \mathrm{Sim}(E_q(x; \theta_q), E_s(s; \theta_s)), \qquad \Phi_\tau(x) = \tau \log \left[\frac{1}{|\mathcal{N}_q|} \sum_{n \in \mathcal{N}_q} \exp(S_\theta(x, n)/\tau) \right], \quad (40)$$

and define

$$R_\tau(x, s) = S_\theta(x, s) - \Phi_\tau(x). \quad (41)$$

Then the single-target loss can be rewritten as

$$\ell(x, s; \mathcal{N}_q) = \log\left[1 + |\mathcal{N}_q| \exp(-R_\tau(x, s)/\tau)\right]. \quad (42)$$

Since $\tau > 0$ and $|\mathcal{N}_q| \geq 1$, the right-hand side is strictly decreasing in $R_\tau(x, s)$. Thus a lower target loss corresponds to a larger relative score for the same negative set. Figure 1 tracks $R_\tau(q, s)$ for the required skills of one query. For a nonempty target set T, a set-level diagnostic is the mean

$$R_\tau(x; T) = \frac{1}{|T|} \sum_{s \in T} R_\tau(x, s). \quad (43)$$

The single-target transformation is nonlinear, so an increase in this mean alone does not establish a decrease in the mean target loss. The score compares a target with the fixed negative candidate set; the Top-K retrieval metrics in Section 5 evaluate rankings over the candidate collection. Training uses the contrastive objectives defined in the main text.

F Skill Graph Construction

SkillComp constructs query-local skill graphs before retriever training. The graphs identify relations among skills associated with each training query and determine which skill descriptions are appended to create expanded queries. The graphs and expanded queries are used only for training.

F.1 Annotation Data

SkillRet provides 63,259 training queries over 10,123 skills (Kang et al., 2026). We annotate relations for its 42,483 multi-skill queries, while the remaining 20,776 single-skill queries use the self-expanded construction described below.

For each query q, let G denote its gold required-skill set. We separate relation annotation into two steps. The LLM first identifies the subset $H \subseteq G$ whose relevance is directly supported by the task description. It then annotates pairwise relations only among skills in H. This avoids assigning a relation to a skill whose role in the task is not sufficiently supported by the query itself. Importantly, H is used only to determine which relations can be reliably annotated. It does not change the retrieval targets: all skills in G, including those outside H, remain positive targets for both the plain and expanded queries. For every supported canonical pair (a, b), the annotator assigns one label $\rho_q(a, b) \in \{\text{A-Before-B}, \text{B-Before-A}, \text{Parallel}\}$. The labels are defined as follows:

- A-Before-B ($a \to b$): a provides an artifact or precondition required by b, b operates on or validates the output of a, or the query explicitly requires a to precede b.
- B-Before-A ($b \to a$): the same dependency holds in the opposite direction.

- PARALLEL ($a \parallel b$): both skills are required, but the query specifies no necessary execution order between them.

Then, we use GPT-5.4-mini for the initial relation annotations. Each request supplies one query, its associated skill identifiers, and the corresponding skill cards, as

Prompt Template for Query-Conditioned Skill-Relation Annotation

Role

You are a conservative, evidence-first query-conditioned skill relation validator.

Inputs

Query: {query}
Supplied skill IDs: {skill_ids}
Skill cards: {skill_cards}

Step 1: Determine skill support

The supplied skills are noisy candidate positives for ONE query. The query, not skill names or candidate order, defines what is required. Return one node for every supplied skill. Never add, remove, merge, or rename IDs.

Mark a skill as supported only if removing its capability leaves an explicit deliverable or explicitly requested check unmet. Generic adjectives or constraints (secure, production-ready, high-quality, maintainable, follows best practices) do not require a separate standards/checker skill unless the query separately asks to review, audit, validate, verify, or demonstrate compliance. Do not promote completed work, background technology, an artifact to inspect, an executor permission, or a keyword match to a required skill.

Step 2: Label every supported pair

For each supported pair, use exactly one canonical record with skill_a_id < skill_b_id.

a_before_b: A must complete before B can fulfil its query role. Require a concrete two-sided bridge: A provides an artifact/result/state/precondition that B consumes; B validates or operates on A's work; or the query explicitly requires A before B. Apply the counterfactual test: if B can fulfil its assigned role without A's result or state, do not choose before.

b_before_a: Apply the same rule in the reverse direction.

parallel: Both skills are required, but no direction is evidenced. This means no ordering constraint, not guaranteed concurrent execution.

Evidence restrictions

Words such as then, after, finally, while, or ensure are not an ordering bridge by themselves. Node roles are descriptive only, never evidence. Do not infer order from input order, common tools, skill-name similarity, generic workflow text, or permissions.

Completeness and graph projection

Return the complete pair matrix, including transitive pairs: if A is before B and B is before C, label A before C as well. Do not output DAG edges or execution stages; deterministic code derives them.

Borderline example

"Implement authentication and ensure production security standards" supports an authentication implementation skill but not a generic code-standards skill, unless review, audit, or validation is separately requested. Use brief source-grounded evidence and no chain of thought.

F.2 RELATION VALIDATION AND GRAPH CONSTRUCTION

Annotation Validation by LLM Review and Manual Audit. We further validate these annotations before using them for training. We first use an independent LLM pass to re-check the annotated relations. The verifier receives the query and candidate skill descriptions without the original labels or rationales and assigns the relation labels again using the same annotation criteria. For annotations with confidence below 0.8, we compare the two decisions and re-annotate cases where they disagree. Relations that remain unresolved are excluded from relation supervision.

We then manually audit 200 retained examples sampled across relation types and confidence ranges. The audit checks both whether each skill is supported by the query and whether the assigned relation follows the annotation criteria. This manual review yields 98% annotation accuracy.

After annotation and validation, the retained data contain 36,523 relations, including 17,549 sequential dependencies and 18,974 co-required relations. The remaining 16,224 multi-skill tasks have no retained relation and therefore use the co-required fallback during training.

Graph-Level Consistency Check. Finally, we perform a graph-level consistency check on the retained relations. Each relation must connect two distinct skills in H, and every pair must receive exactly one relation label. We reject annotations with missing or duplicate pairs, endpoints outside H, conflicting directions, dependency cycles, or inconsistent relations. For example, if a precedes b and b precedes c, labeling (a, c) as PARALLEL is inconsistent.

The validated pairwise relations are then converted into a task-specific skill graph. Directional relations define sequential dependencies, while PARALLEL relations define co-required pairs. We apply transitive reduction to the directional relations to retain only direct dependencies. For example, if the pairwise annotations contain $a \to b$, $b \to c$, and $a \to c$, the transitive relation $a \to c$ is removed, leaving $a \to b \to c$. The resulting graph is $\mathcal{G}_q = (H, D_q, P_q)$, where $D_q \subseteq H \times H$ contains direct sequential dependencies and $P_q \subseteq \binom{H}{2}$ contains co-required pairs.

Selection for co-required pairs. For each co-required pair, we select the endpoint with lower similarity to the plain query under the current retriever. The selected skills and hard negatives are refreshed every 50 training steps.

Expanded queries. Each task receives the expanded query $q_r = q \oplus r$ for its selected skill r; a single-skill task appends its sole required skill. For a three-skill task whose skills form a sequential workflow, a second expanded query $q \oplus r \oplus r'$ further appends a skill r' adjacent to r in the dependency graph. Thus, $\mathcal{A}_q = \{\{r\}\}$ or $\{\{r\}, \{r, r'\}\}$.

Annotation Statistics. The 26,259 queries with retained relations form the graph structures summarized in Table 12. This table counts task-specific graphs, whereas Table 12 counts the 36,523 annotated skill pairs before transitive reduction. For example, a three-skill total order contains three pairwise directional relations, $a \to b$, $b \to c$, and $a \to c$, but its reduced graph contains only the two direct dependencies $a \to b$ and $b \to c$.

Table 12: Structures of the 26,259 retained local skill graphs.

Gold-set size	Retained graph structure	Count
Two skills	One sequential dependency	5,666
	One co-required relation	5,570
Three skills	One sequential dependency	5,211
	One co-required relation	4,680
	Three co-required relations	1,709
	One dependency and two co-required relations	1,265
	Branching: $a \to b$, $a \to c$, $b \parallel c$	397
	Converging: $a \to c \leftarrow b$, $a \parallel b$	670
	Total order: $a \to b \to c$	1,091
All retained query-local graphs		**26,259**

F.3 Illustrative Construction

Illustrative Construction of an Expanded Query View

Task
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?

Original gold skills
mermaid-master. World-class creation of valid, beautiful, and accessible Mermaid diagrams. Use when users request diagrams, flowcharts, sequence diagrams, entity relationship diagrams, state machines, Gantt charts, class diagrams, or any visual representation that can be expressed in Mermaid syntax. Handles all Mermaid diagram types with expert knowledge of syntax, styling, and best practices.
text-to-speech. Converts text to speech audio using OpenAI TTS API. Use when users request audio versions of text or want responses read aloud.

Relation Both skills are supported by explicit deliverables. The diagram skill produces the text consumed by the speech skill, giving the retained relation mermaid-master $\rightarrow$ text-to-speech with confidence 0.96.

Training views
Plain view q: the original task. Positive target G_q: mermaid-master; text-to-speech.
Expanded query q_r: the original task with the mermaid-master description appended. Positive target G_q: mermaid-master; text-to-speech.

F.4 CASE STUDY

(a) Adaptive Cruise Control

Query:

You need to implement an Adaptive Cruise Control (ACC) simulation that maintains the set speed (30m/s) when no vehicles are detected ahead, and automatically adjusts speed to maintain a safe following distance when a vehicle is detected ahead. The targets are: speed rise time <10s, speed overshoot <5%, speed steady-state error <0.5 m/s, distance steady-state error <2m, minimum distance >5m, and control duration 150s. The constraints are: initial speed ~0 m/s, acceleration limits [-8.0, 3.0] m/s², time headway 1.5s, minimum gap 10.0m, emergency TTC threshold 3.0s, and timestep 0.1s. Data is available in vehicle_params.yaml (vehicle specs and ACC settings) and sensor_data.csv (1501 rows with time, ego_speed, lead_speed, and distance collected from real-world driving). Create pid_controller.py, acc_system.py, and simulation.py; tune speed and distance PID parameters into tuning_results.yaml, then run the 150s simulation to produce simulation_results.csv and acc_report.md.

Required skill	Description	Qwen-Embedding-0.6B	SkillRouter	SkillRet	R3-skill	Ours
G1 csv-processing	reading sensor data from CSV files, writing simulation results to CSV	13,664	81	178	22	**3**
G2 pid-controller	implementing PID control loops for adaptive cruise control, vehicle speed regulation	2	2	1	2	**1**
G3 simulation-metrics	calculating control system performance metrics such as rise time, overshoot percentage, steady-state error	6	3	4	5	**5**
G4 vehicle-dynamics	simulating vehicle motion, calculating safe following distances, time-to-collision, speed/position updates	1	1	2	1	**2**
G5 yaml-config	reading or writing YAML configuration files, loading vehicle parameters	34,938	347	1,339	76	**10**
Completeness@20		3/5	3/5	3/5	3/5	**5/5**

(b) Spring Boot Migration

Query:

Under the /workspace/ folder, there is a legacy user management microservice that you need to migrate from Java 8/Spring Boot 2.7 to Java 21/Spring Boot 3.2. This microservice is based on REST API and manages user information. Key functions are: CRUD operations for user management, JWT-based authentication, Role-based access control, Database persistence with JPA/Hibernate, Input validation, and External API calls using RestTemplate. Your migration should upgrade Java version from 8 to 21, Spring Boot version from 2.7.x to 3.2.x and upgrade all related dependencies to compatible versions. Make sure the namespace migration is properly handled. You also need to ensure Hibernate 6 compatibility, update to Spring Security 6, migrate RestTemplate to RestClient, and fix deprecated security methods. Make sure to run mvn clean compile and mvn test to check the quality.

Required skill	Description	Qwen-Embedding-0.6B	SkillRouter	SkillRet	R3-skill	Ours
G1 hibernate-upgrade	Migrate Hibernate 5 to Hibernate 6 with Spring Boot 3.	88	4	39	15	**3**
G2 jakarta-namespace	Migrate Java EE javax.* imports to Jakarta EE jakarta.* namespace.	18	3	29	9	**4**
G3 restclient-migration	Migrate RestTemplate to RestClient in Spring Boot 3.2+.	61	5	81	5	**1**
G4 spring-boot-migration	Migrate Spring Boot 2.x applications to Spring Boot 3.x.	3	2	1	8	**5**
G5 · spring-security	Migrate Spring Security 5 to Spring Security 6 configuration.	42	39	33	33	**20**
Completeness@20		2/5	4/5	1/5	4/5	**5/5**

Figure 7: Case study.